Proceedings for Wide Area Information Server Workshop

February 3-4, 1992
MCNC
Research Triangle Park, NC USA

Sponsored by
MCNC Center for Communications,
the National Science Foundation and
the Institute for Academic Technology

Conference Chairperson

George Brett, MCNC Center for Communications

Conference Co-chairperson

Brewster Kahle, Thinking Machines Corporation

John Oberlin, Institute for Academic Technology

PROCEEDINGS

WIDE AREA INFORMATION SERVER WORKSHOP

FEBRUARY 3-4, 1992
RESEARCH TRIANGLE PARK, NORTH CAROLINA, USA

SPONSORED BY MCNC CENTER for COMMUNICATIONS,
the NATIONAL SCIENCE FOUNDATION
AND
the INSTITUTE OF ACADEMIC TECHNOLOGY

FINAL PROGRAM

Monday, February 3, 1992 - MCNC

9:00 - 10:00 a.m.	**Registration/Continental Breakfast**
10:00 - 11:00 a.m.	**Introductions and Overview** G. Brett, MCNC Center for Communications A. Blatecky, MCNC Center for Communications *MCNC Introduction and Workshop Overview.*
11:00 a.m. - Noon	**Session I - Introduction to Wide Area Information Servers** B. Kahle, Thinking Machines Corporation *Introduction to and background on the Wide Area Information Server.*
Noon - 1:00 p.m.	**Lunch**
1:00 - 2:00 p.m.	**Session I - Introduction to Wide Area Information Servers *(continued)*** B. Kahle, Thinking Machines Corporation
2:00 - 2:30 p.m.	**Break**
2:30 - 4:00 p.m.	**Session II - Z39.50 Protocol and Wide Area Information Servers** C. Lynch, University of California *Presentation of Z39.50 protocol and its relationship to WAIS.*
4:00 - 5:00 p.m.	**Demonstrations of WAIS clients and servers**
6:30 - 8:00 p.m.	**Reception - Hors d'oeuvres at the Institute for Academic Technology, Research Triangle Park**

Tuesday, February 4, 1992 - MCNC

8:00 - 8:30 a.m.	**Continental Breakfast**
8:30 - 11:00 a.m	**Session III - Wide Area Information Server Development at UNC-Chapel Hill** J. Fullton, Office of Information Technology, UNC-Chapel Hill *Emphasis will be on current and projected development of WAIS clients, servers, and databases.*
11:00 - 11:30 a.m.	**Break and Demonstrations**
11:30 a.m. - Noon	**Session IV - The Kudzu Project in North Carolina** G. Brett, MCNC Center for Communications *This session will summarize the workshop and outline future activities.*
Noon - 1:00 p.m.	**Adjourn to Lunch**

Introduction and Overview

G.Brett and A. Blatecky

Welcome

George H. Brett II, MCNC & UNC-General Administration

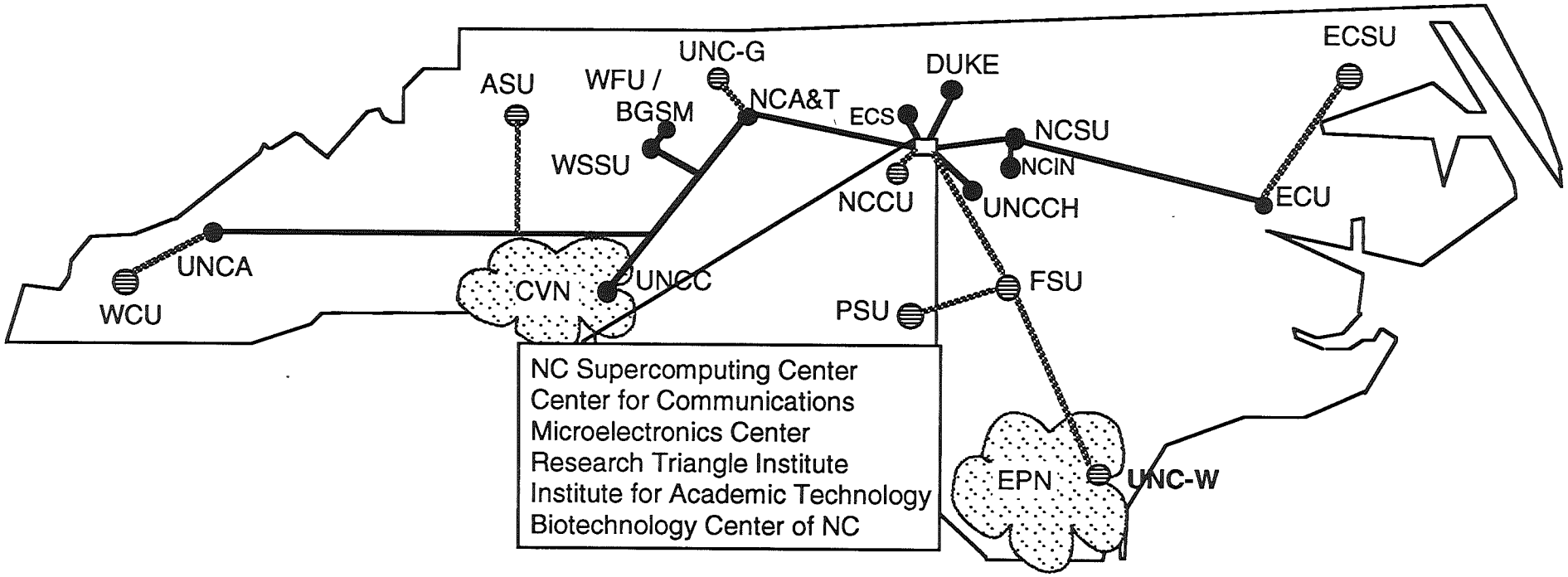
This two-day conference launches the North Carolina WAIS Initiative by gathering people who have worked on the WAIS program and people who would like to understand how WAIS can offer an electronic publishing backbone for network information distribution.

- Speakers will include people who have created the tools, and topics will focus on where their efforts are going.
- The conference will include demonstrations of all public interfaces and services available on WAIS.
- Both industry and academic projects will be presented.
- Users of the technology will discuss how it already has affected their organizations and what their plans are for future development.

Wide Area Information Server Workshop

Center for Communications

CONCERT

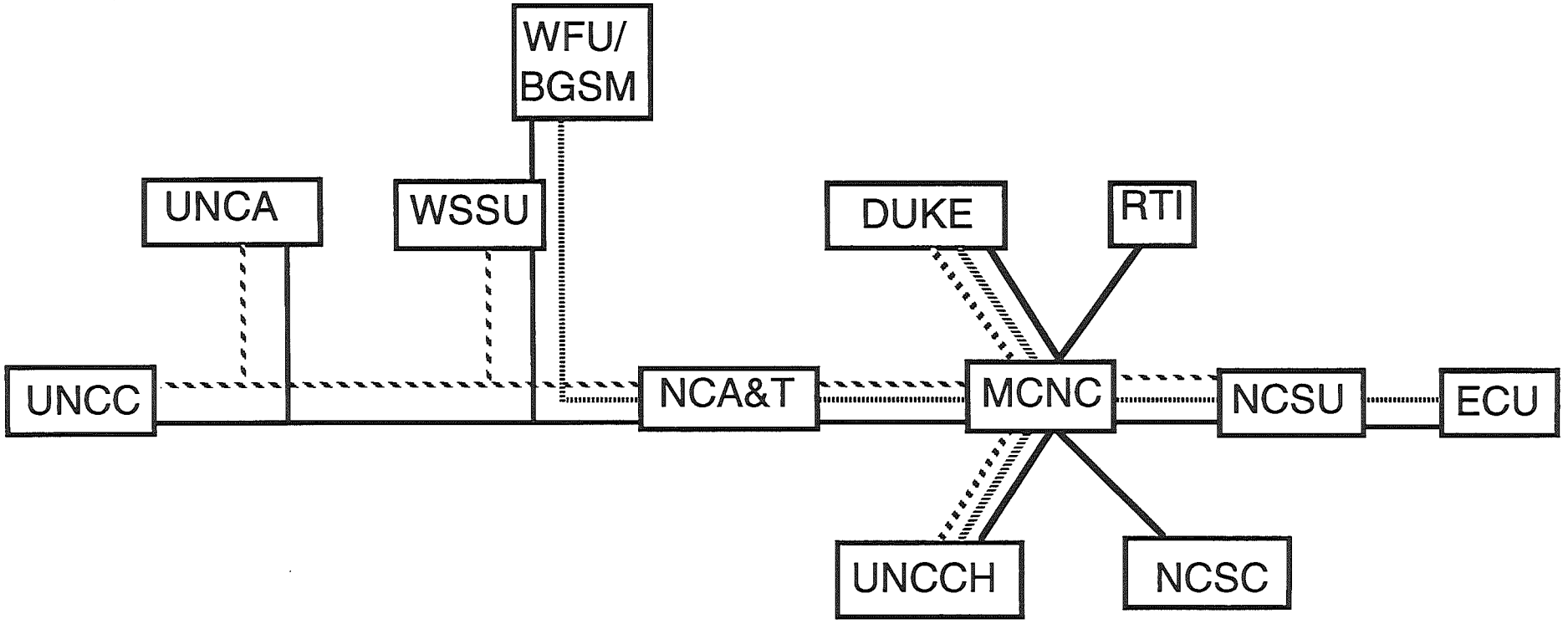


Potential future connections

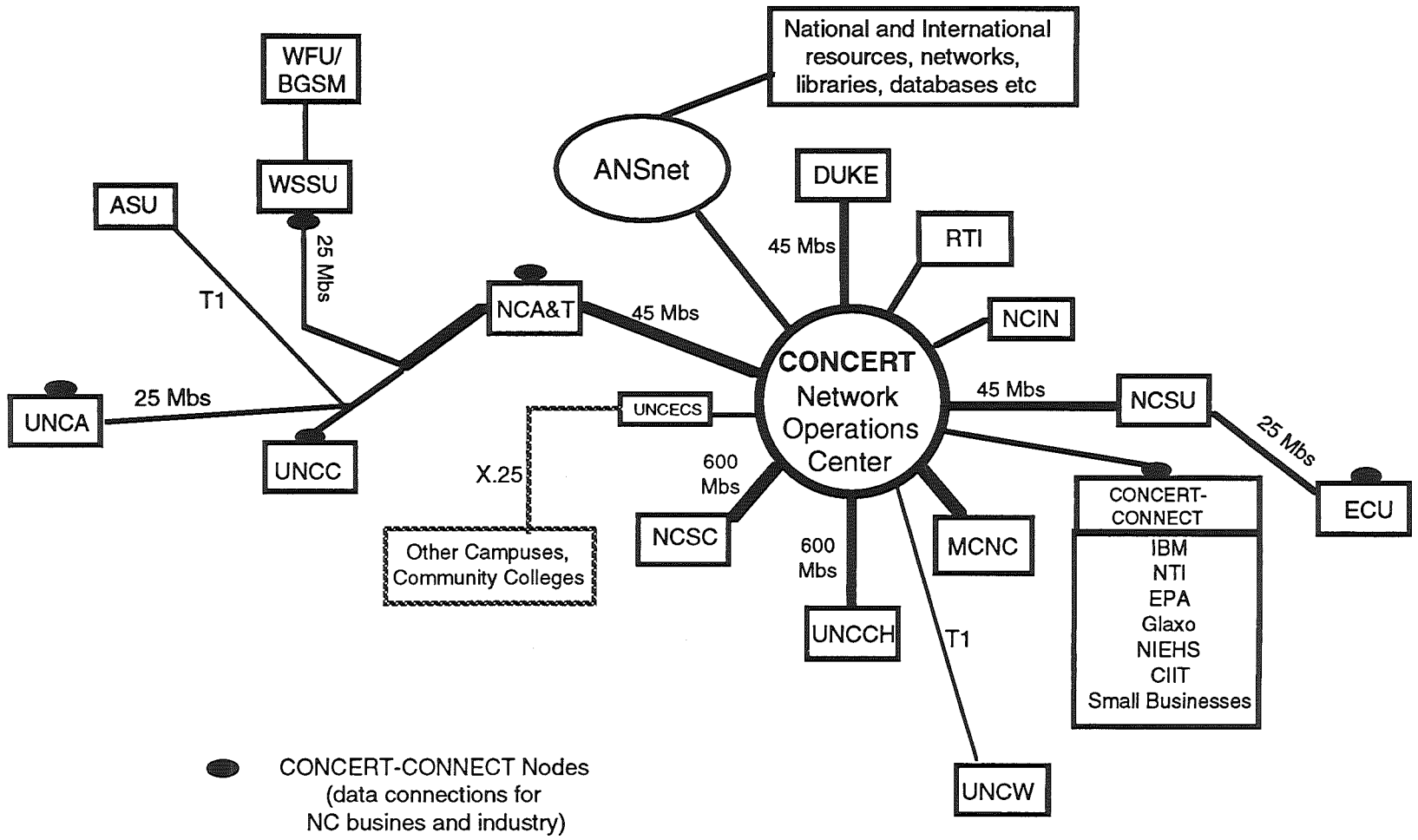
BellSouth Educational Network Initiatives

Virtual Proximity

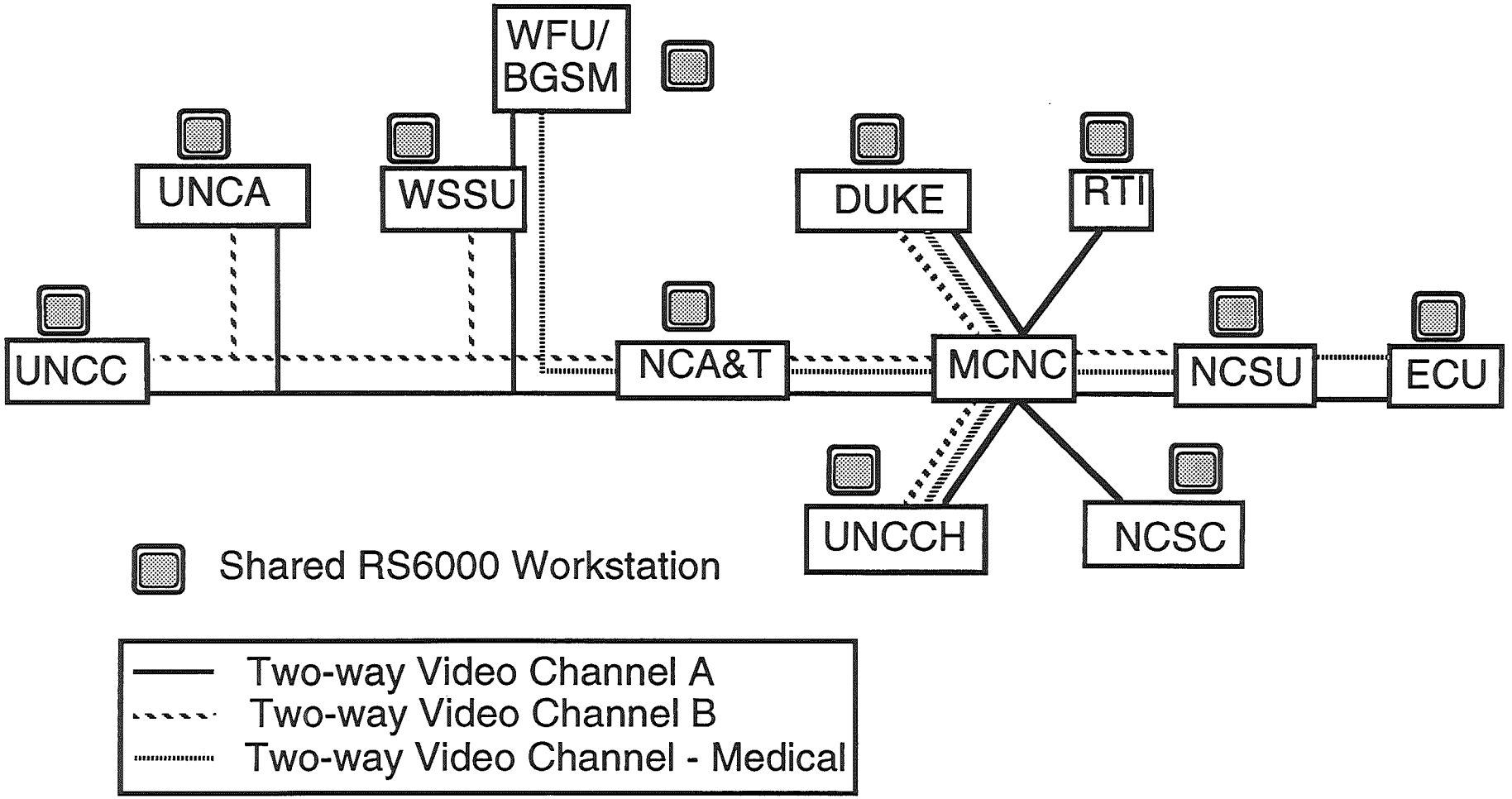
CONCERT Video Network



CONCERT Data Network

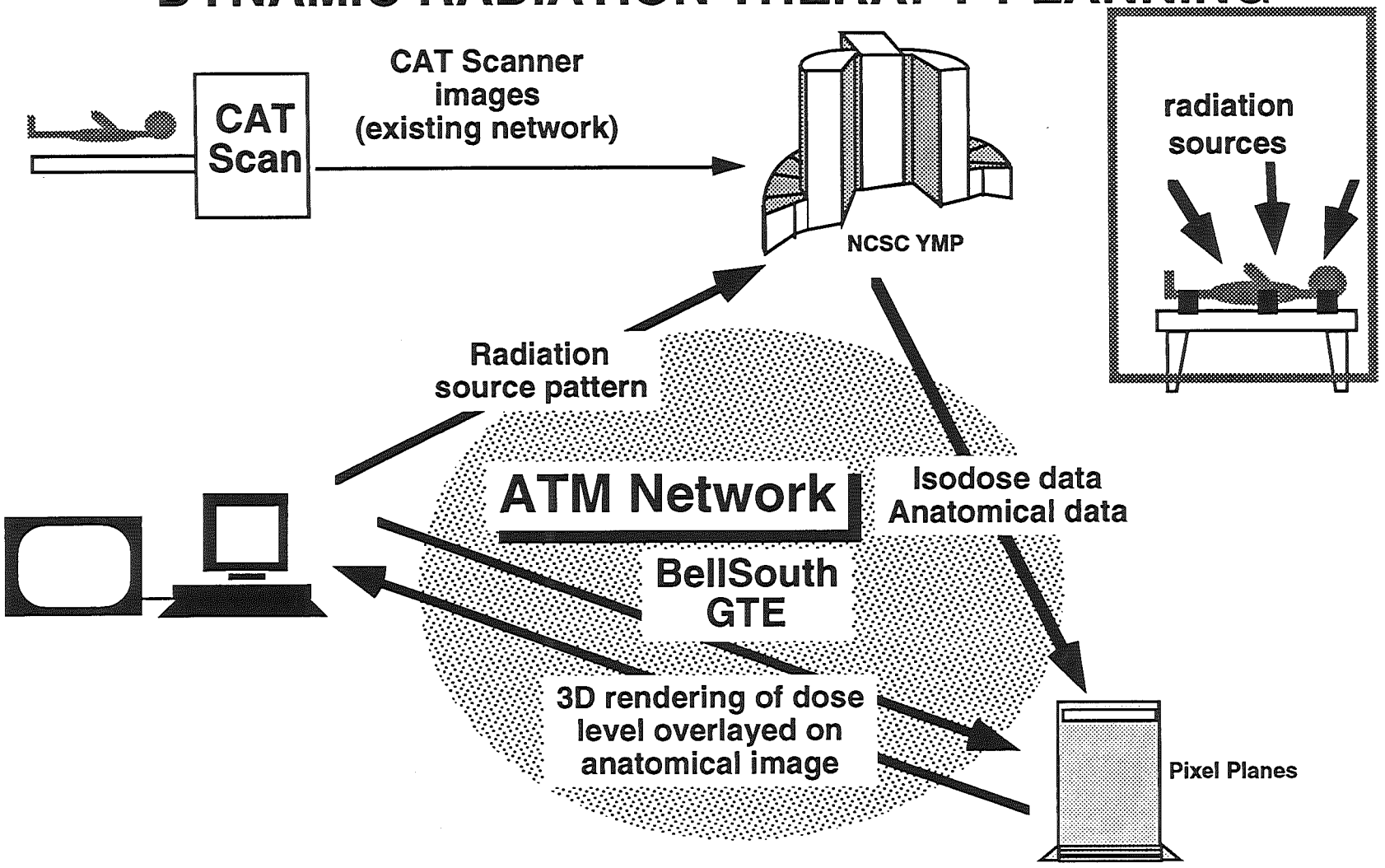


Shared Workstation Collaboratory Project



VISTAnet Summary

DYNAMIC RADIATION THERAPY PLANNING



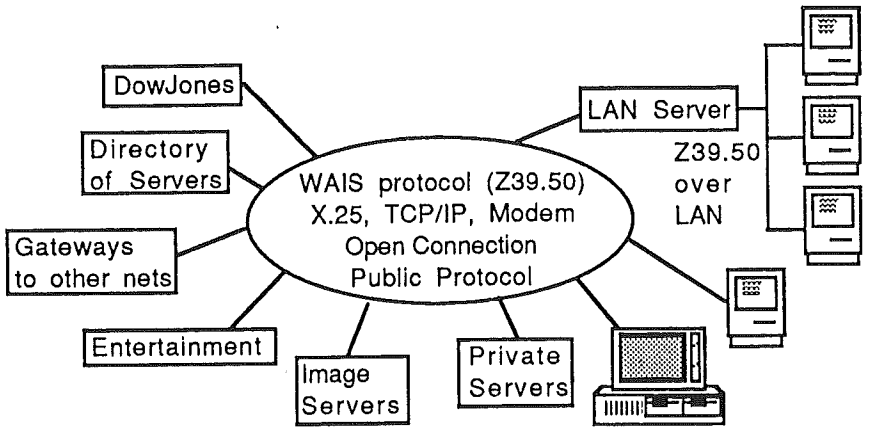
Session I: Introduction to Wide Area Information Servers

B. Kahle

Wide Area Information Servers

Brewster Kahle

December 1991



Users Needs:
- Selecting Servers
- Answering Questions
- Organizing Responses

Architecture Issues:
- Scalability
- Security
- Business model for servers
- Reliable Access

The Wide Area Information Servers (WAIS) system is an electronic publishing system that helps end users find unstructured information located on remote machines. It is composed of user interfaces, available for most machines, and server software. Started by Thinking Machines, this system is becoming a standard for information distribution in the internet environment. Since many components are available for free, please try the system!

What does WAIS do? Users on different platforms can access personal, company, and published information from one interface. The information can be anything: text, pictures, voice, or formatted documents. Since a single computer-to-computer protocol is used, information can be stored anywhere on different types of machines. Anyone can use this system since it uses natural language questions to find relevant documents. Relevant documents can be fed back to a server to refine the search. This avoids complicated query languages and vendor specific systems. Successful searches can be automatically run to alert the user when new information becomes available.

REGIONAL OFFICE:
1010 EL CAMINO REAL, SUITE 310
MENLO PARK, CA 94025
TELEPHONE: (415) 329-9300
FAX: (415) 329-9329

CORPORATE HEADQUARTERS:
245 FIRST STREET
CAMBRIDGE, MA 02142-1264
TELEPHONE: (617) 234-1000
FAX: (617) 234-4444

How does WAIS work? The servers take a users question and do their best to find relevant documents. The servers, at this point, do not "understand" the users English language question, rather they try to find documents that contain those words and phrases and ranks then based on heuristics. The user interfaces (clients) talk to the servers using an extension to a standard protocol Z39.50. Using a public standard allows vendors to compete with each other, while bypassing the usual proprietary protocol period that slows development. Thinking Machines is giving away an implementation of this standard to help vendors develop clients and servers.

What WAIS servers exist? Even though the system is very new, there are already over 100 servers on the internet. Over 5000 people have used WAIS in 20 countries.

- Thinking Machines operates a Connection Machine on the internet for free use. The databases it supports are some patents, a collection of molecular biology abstracts, a cookbook, and the *CIA World Factbook.*
- MIT supports a poetry server with a great deal of classical and modern poetry. Cosmic is serving descriptions of government software packages. The Library of Congress has plans to make their catalog available on the protocol.
- Weather maps and forecasts are made available by Thinking Machines as a repackaging of existing information.
- The "directory of servers" facility is operated by Thinking Machines so that new servers can be easily registered as either for-pay or for-free servers and users can find out about these services.
- Dow Jones is putting a server on their own DowVision network. This server contains the *Wall Street Journal, Barrons*, and 450 magazines. This is a for-pay server.

How can I find out more about WAIS?

- You can try a simple interface by telneting to quake.think.com, login wais.
- FTP the free software from think.com in the /wais directory.
- FTP a bibliography:
/pub/wais/wais-discussion/bibliography.txt@quake.think.com
- Contact Barbara Lincoln (barbara@think.com) for more information, or Brewster Kahle the project leader.
- Subscribe to a biweekly mailing list on electronic publishing issues, and new releases; to subscribe send email to wais-discussion-request@think.com.

Brewster Kahle
Project Leader
Wide Area Information Servers
Brewster@Think.com

Wide Area Information Servers: A Supercomputer on every Desk

Brewster Kahle
Thinking Machines Corporation

What I really want...

- My personal information to be accessible
- Published information should find me
- Usable anywhere
- Others can use what I have learned (if I want them to)

What is it?

Electronic Publishing

(Or publishing over wires)

New Communications Technology Problems

	BOOKS	Telegraph> Telephone	Electronic Publishing
Experts only	*Monks*	*Operators*	*Professional searchers*
Distribution is hard and expensive	*Vellum is calf skin*	*Telephones on barb wire*	*$1/minute over obscure modems*
Different interfaces	*1000's of languages in Europe alone*	*Switching was manual*	*//query (W5) inform?*
Material is intractable	*Scrolls and manu-scripts were about as random access as musical scores*	*No white pages*	*600 databases on Dialog ~1 Terabyte 140Gbyte at DJ 80GB card catalog at RLG*
Business model needed	*Centralized printing*	*Pay per minute*	*Not understood*

Navigation Techniques: Paper

- Alphabetical Listings (dictionary, Encyclopedia)
- Indices (back of the book and Readers Guide)
- Table of Contents (outlining)
- Citation index
- "Tree of Knowledge"
- Have you read any good books lately?

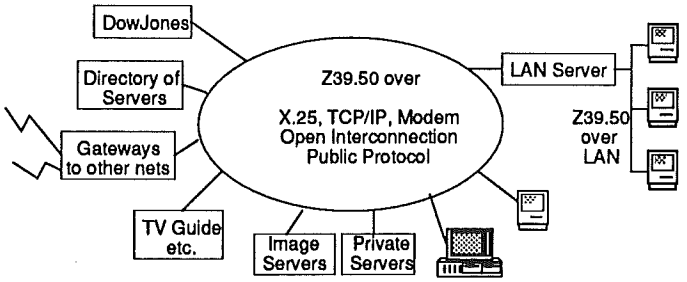
Navigation Techniques: Computers

- Hierarchical File Systems
- Unix "find" and "grep", Mac "find file"
- Boolean query systems (...within 5 words of...)
- Static Hypertext links (see also pointers)

Navigation Techniques: WAIS

- English language questions and Relevance feedback
  * Iterative retrieval
  * Question-answer dialog
  * Similar to the Newspapers front page the: "continued on page 5"
  * Dynamic Hypertext Links
- 2 level search:
  * Directory of servers (server like any other)
  * Servers themselves
- Copy editors help select documents
  * Easy to "publish" opinions on documents

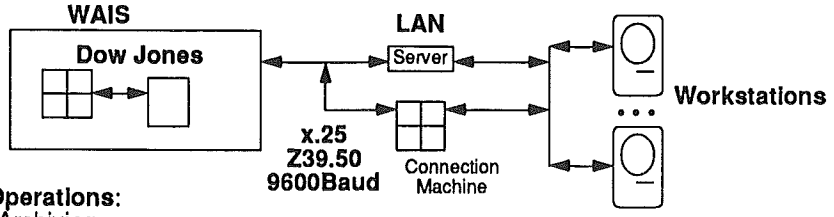
Wide Area Information Server Architecture



Users Needs:
- **Selecting Servers**
- **Answering Questions**
- **Organizing Responses**

Architecture Issues:
- **Scalability**
- **Security**
- **Business model for servers**
- **Reliable Access**

Demonstration System Structure



Operations:
Archiving
Queries
Retrieval
IR Type:
Broadcast
Query by Example
Databases:
Wall St Journal
Barron's
400 Business Mags

CM: Operations: Queries
IR Type:
enhanced relevance feedback
DBs: DowVision and
memo's, mail,
word processor files

Mac:
Operations:
Human Int
Retrieval
Queries
"Caching" Docs
User Profiles
IR Type:
Query by example
DBs:
Personal Text
Cached data

WAIS Clients

- Busy 24 hours a day finding information
- Ponder all indications of the preferences of its user
- Gossip with other clients about their discoveries
- Scours the world (within a budget) to find new sources

WAIS Protocol

- Based on Z39.50, bypass proprietary period
- Flexible
- Non Threatening for corporations

- Search: (words, doc_ids, databases) -> server returns list of: (headline, score, doc_id, types)'s
- Retrieval: (doc_id, type, start, end) -> server returns: bunch of bytes

- Doc_id: An ISBN for the Electronic Age
  ((orig_server, orig_database, orig_local_id)
  (dist_server, dist_database, dist_local_id)
- Server Description:
  (:ip-address, :database-name, :cost, :description)

Connection Machine Server

- 1-25GBytes (and getting bigger)
- Supports thousands of users
- Automatic Indexing
- Uses words and phrases in question to find appropriate documents
- First turn-key massively parallel application

TMC Internet Release

- CM product for TCP/IP (complete server)
- Example User interfaces for free (no support)
  Macintosh, Gnu Emacs, Xwindows
- Example unix server software to create servers
- Directory of Servers on the internet at least through '91
- 42 Servers now: Weather Maps, patents, Government programs, Risks-digest, usenet recipies, Lewis Carroll,...
- Anonymous FTP Think.com:/public/wais/*
  Mailing list: wais-discussion-request@think.com

Conclusion

- Electronic Publishing can fill niches now
- Companies are positioning themselves now (workstations, server, and info providers)
- Thinking Machines is the "Engine of the Information Industry"

UNIX Today! December 3, 1991

The Promise Of The WAIS Protocol

Emerging Standard Represents First Step Toward Unifying Data Search & Retrieval

BY JASON LEVITT

It doesn't take an expert to see that the state of modern information handling is neither open nor unified. A trip to the main library at the University of Texas at Austin—one of the top 10 college library systems in the U.S.—confirms this.

The primary card catalog is contained on an IBM mainframe accessible through various synchronous block-mode terminals scattered about the main library, and also accessible via modem through a rather crude dial-up facility.

In the main reference room, an OCLC (On-line Computer Library Center) terminal allows access to other university card catalogs; several IBM PCs are available to search CD-ROMs for bibliographical citations and abstracts on a variety of subjects; and a LEXIS/NEXUS terminal can be used for researching major U.S. court decisions. In the engineering library, an IBM PC with CD-ROM is available for searching U.S. patents.

WAIS Server Source File

```
(  :source
   :version 3
   :ip-name "nextbox.utoday.com"
   :tcp-port 5001
   :database-name "UT_TECH"
   :cost 0.00
   :cost-unit :free
   :maintainer "jason@nextbox.utoday.com"
   :description "Server created with WAIS release
   8 b2 on Mon Nov 18 16:54:19 1991 by
   jason@nextbox.utoday.com
   UNIX Today! technology articles by Jason Levitt
   The files of type text used in the index were:
      /LocalLibrary/WAIS/articles/ABCstory.txt
      /LocalLibrary/WAIS/articles/AIX3.1FS.txt
      /LocalLibrary/WAIS/articles/Benchinfo.txt
      /LocalLibrary/WAIS/articles/LPFstory.txt
      /LocalLibrary/WAIS/articles/MacXstory.txt
      /LocalLibrary/WAIS/articles/Solbourne.txt
      /LocalLibrary/WAIS/articles/SunStory.txt
      /LocalLibrary/WAIS/articles/XSerialArticle.txt
      /LocalLibrary/WAIS/articles/Xarticle.txt
      /LocalLibrary/WAIS/articles/Xcontrib.txt
)
```

Figure 1

If one were to compare information accessibility at this facility to computer resource accessibility, things here are still in the early 1980s or late '70s. Each of the systems mentioned are primarily standalone and proprietary, having their own information retrieval and organizational formats with little, if any, interoperability between databases.

While the monster mainframe card catalog might provide pointers to many sources, it is ignorant of most other on-line sources and almost never provides the most current information on subjects, despite the best efforts of its administrators. What these information-handling systems need is a dose of open systems standards and technology, the same technology that is changing the face of modern computing.

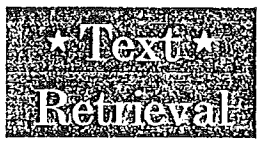
Enter WAIS, for Wide-Area Information Server, a fledgling step in the overwhelming effort needed to unify information search and retrieval technology. WAIS is an emerging open systems standard protocol for query and retrieval of information. WAIS, pronounced "ways," is the brainchild of Brewster Kahle, an employee of Thinking Machines Corp. (TMC), the No. 2 supercomputer manufacturer, behind Cray, and purveyor of fine, massively parallel systems.

The basis for WAIS is the rapidly growing electronic-publishing movement, which is seeing more and more materials, usually available only in book form, "published" or placed onto electronic media such as disk and tape, where it can be accessed with a computer.

WAIS TECHNOLOGY

WAIS is a protocol for the transmission of query and retrieval information, much like the information you would use to search a library card catalog. It is, in fact, an extension to an existing protocol standard called Z39.50, the Information Retrieval Service Definitions and Protocol Specification for Library Applications.

The Z39.50 standard was created by a group called NISO, the National Information Standards Organization, and is designed for use in electronic library card catalogs. Z39.50 essentially specifies formats for search requests directed at a database and formats for document retrieval requests. WAIS extends the Z39.50 standard to allow, among other things, discrete portions of documents, called "chunks," to be retrieved. This is especially useful in low-bandwidth situations such as serial links, where transferring an entire document in response to a query would be prohibitively time-consuming.



The WAIS protocol fits neatly at the top of the ISO 7-layer protocol model at the application and presentation layers. This makes it extremely portable to differing network environments such as TCP/IP and X.25.

Like any good open standard, the WAIS protocol does not specify or limit the technology at either end of the wire. A WAIS client can be as simple as a command line interface that takes a database name, network address and query string as input, or as complex as a combination spreadsheet and database that constantly updates in real time, based on client/server activity taking place in the background. The only condition is that the client and server exchange query and retrieval information using the WAIS protocol.

The free WAIS source code, discussed later, implements a very typical client/server model for Unix-based Internet applications. The server creates and waits on a socket attached to a well-known port. Clients attach to the port using the port number and network address of the machine. The server accepts a request, forks a child process to handle the request, and then continues to wait and service other requests.

Requests for information are largely governed by special text files maintained by the WAIS server, called "sources," that vaguely resemble library catalog cards. Figure 1 shows a source I created containing 10 of my previous technology articles for *UNIX Today!* There is enough information in the source structure, network address, TCP port number and database name for any other machine on the network running a WAIS client to locate, understand and access the information in the database.

Not surprisingly, WAIS is already being used to connect archive sites on the Internet running on various Unix-based machines as well as proprietary systems such as Macintosh and NeXT. According to Brewster Kahle, there are approximately 80 sites running public WAIS servers and many more running WAIS privately within corporations and academia. A FidoNet WAIS server site was recently added to this collection of public sites running SLIP over a 9,600-bps serial link.

FREE WAIS SOFTWARE

I like software that you can use to get some meaningful work done quickly without having to dig too deeply into documentation. The freely available WAIS software fits that description. In the *UNIX Today!* labs, I decided to put together a small heterogeneous network and run WAIS.

Acting as the WAIS server system (and also a client) was a NeXTstation. Attached over Ethernet was a Macintosh running MacOS and a Sun 3/60 running SunOS 4.1. The free WAIS software included NeXT and Mac binaries and complete source code for the Unix systems, in this case the Sun. I dug out my archives of personal Unix electronic mail, about 10 Mbytes' worth, and used the indexing program included with the WAIS server to create a hashed database. I did the same with 10 of my old technology articles written for *UNIX Today!* The databases, or "sources," are listed in Figure 2.

The WAIS indexing program knows about the format of many common types of structured online data such as electronic mail, *netnews*, PICT-/GIF/TIFF files and biology abstract formats, and it also handles straight ASCII text.

There was also a database of WAIS documentation, created automatically by the server program, and a directory of all sources I created called "directory of information" that simply points to all the databases. After creating the databases, I ran the WAIS server program, called *waisserver*, on the NeXTstation, which sits and waits for incoming WAIS client requests.

Once the *waisserver* was running, I could access it using the clients, called WAISstations. On the Sun, which was running X/Motif, I chose to use the Motif client. I also used the Mac and NeXT WAISstations. In order to access a *waisserver*, I first had to set up my sources. Figure 3 shows a source setup window for the Mac client. I had named my database of articles "UT-TECH" on the WAIS server. The access method, "Contact," was MacTCP, Apple's TCP/IP

Continued on page 47

General WAIS Information

Thinking Machines Corp.
1010 El Camino Real, Ste. 310
Menlo Park, CA 94025
415-329-9300 Fax: 415-329-9329
Bibliography of available WAIS documents.
Send electronic mail to: barbara@think.com

Accessing a WAIS client on the Internet
Telnet to quake.think.com, login as **wais**

Getting involved with the Nat'l Public Network

Electronic Frontier Foundation
155 Second Street
Cambridge, MA 02141
617-864-0665
E-mail: eff@eff.org

On-Line WAIS Discussions And Development

alt.wais newsgroup on USENET
Join mailing lists by sending e-mail to:
wais-discussion-request@think.com – Weekly digest of mail from users and developers
wais-interest-request@think.com – Infrequent announcements of new releases
wais-talk-request@think.com – Developers' mailing list

Free WAIS client software
Clients for NeXT; X, Macintosh, Unix ASCII, GNU Emacs and Motif.
Anonymous FTP to think.com in the directory /wais

Clients for VMS, MS-DOS, Novell LAN Workplace and SunView.
Anonymous FTP to samba.oit.unc.edu in the directory /pub/wais/UNC

Free WAIS server software
Servers for NeXT and various Unix platforms
Anonymous FTP to think.com in the directory /wais

Focus On WAIS

Continued from page 44

protocol stack.

As shown in Figure 2, I decided to search my mail archives and technology articles for references to NCD's Xremote protocol. The results appear in the scrolling list. If the result is an entire file, such as the article contained in the file "XSerialArticle.txt," the path name for the file is listed after it.

The other results in the list are individual E-mail messages that actually are in several large text files on the WAIS server. Because the WAIS indexing program understands E-mail format, it was able to index individual E-mail messages in my E-mail archive files and transfer only those E-mail messages pertinent to the client query.

By clicking on a document in the Results window, the portion of the result most relevant to my query appears in another window. The *waisserver* uses a simplistic approach to interpreting my request for information about Xremote. It looks for the word "xremote"—the search is case-insensitive—in mail messages and headers and displays matching documents and mail messages in the results window. This turns out to be adequate as long as you put meaningful words in your query.

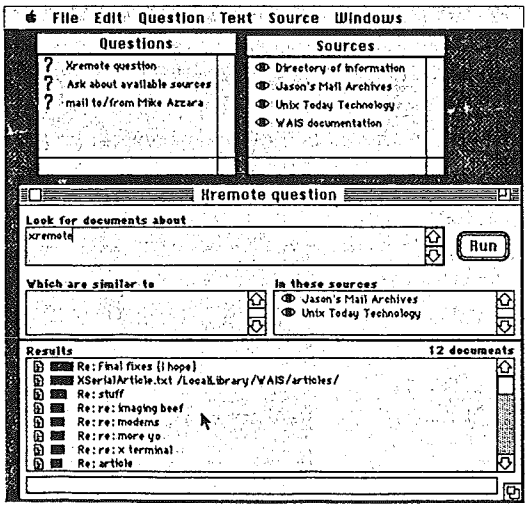


Figure 2: Mac WAIS client shown with results of a search for "xremote"

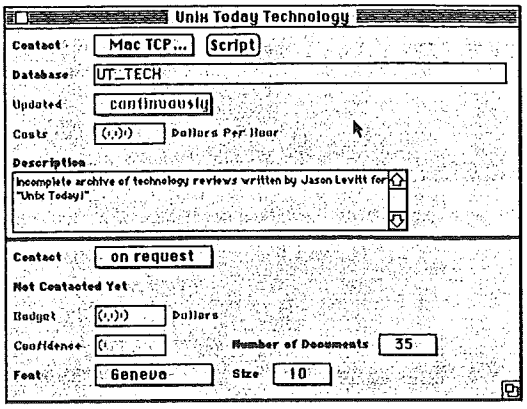


Figure 3: WAIS client set up to use Mac TCP/IP and the database UT_TECH

TMC has a much more sophisticated searching mechanism in its Internet server, *quake.think.com*; however, the search source code is not freely available.

One of the key features of the WAIS protocol is its ability to allow secondary search criteria. In Figure 2, the criteria would be entered by copying a result, or chunk of a result, to the "which are similar to" window. A subsequent search would use any words contained in that window as additional search criteria. Repeatedly using that method can quickly refine the search parameters.

AN OPEN END

The next version of the WAIS protocol should be officially folded into the Z39.50 standard this month and is expected to include multimedia support and integral support for English-language queries. These enhancements should add considerable clout to WAIS, given the infant state of commercial multimedia query/retrieval technology.

WAIS software is freely available from a number of sites. Unfortunately, the WAIS client program can only be obtained via anonymous FTP at this time, which means you have to have direct Internet access.

The WAIS server and X-based client program for Unix are available on *uunet.uu.net* in the directory */networking/distrib-is/wais*.

My small network experiment with WAIS only touched on its full potential; however, for my small database needs, it was quite useful. The free WAIS software is, like the MIT X software, meant as reference software for further development, not as a commercial-quality implementation.

Continued on page 48

Spotlight On WAIS

Continued from page 47

I encountered bugs, such as a persistent permissions error from the NeXT client, and strange window clipping from the Motif client, and I have yet to get the *waisserver* running cleanly under SVR4. But when the software is open and free, who cares?

The vision of WAIS is not only easy access, retrieval and publishing of information, but the creation of a marketplace that can encourage new information sources.

That, according to advocacy groups such as the Electronic Frontier Foundation, could be realized through ISDN, an infrastructure for a "National Public Network" that already is partially implemented in the U.S. telephone system. Such a network could bring the reality of WAIS-based online information services into virtually every home.

Business Day

The New York Times

For Shakespeare, Just Log On

Large PC Libraries Are Being Developed

By JOHN MARKOFF

The development of a nationwide data network will allow personal computer users to tap sources as large as the Library of Congress or receive their own personalized electronic newspapers.

Several innovations, taken together, have already demonstrated that searching vast computer data bases can be easier than consulting a card catalogue, and not nearly as difficult or expensive as computer searches are today. Computer users might read some Dickens more readily than they could check out David Copperfield from the local library.

Those in the industry say that users with little computer skills will soon be able to search through several terabytes of information, or several trillion characters of text, in seconds. The Library of Congress, with 80 million items, contains an estimated 25 terabytes of information.

Already, an experimental computer library has linked 150 universities to 40 sources of information, ranging from National Institutes of Health data to corporate documents and Shakespeare's plays. New software allows users to browse or zero in on particular information.

As methods of retrieving information are standardized and perfected, industry executives and computer scientists say, thousands of new services, ranging from electronic newspapers to the computer equivalent of free public libraries, will blossom. "Everyone is realizing how important it is to get into the mass market for information," said Thomas Koulopoulos, president of Delphi Consulting Group, a Boston market research firm.

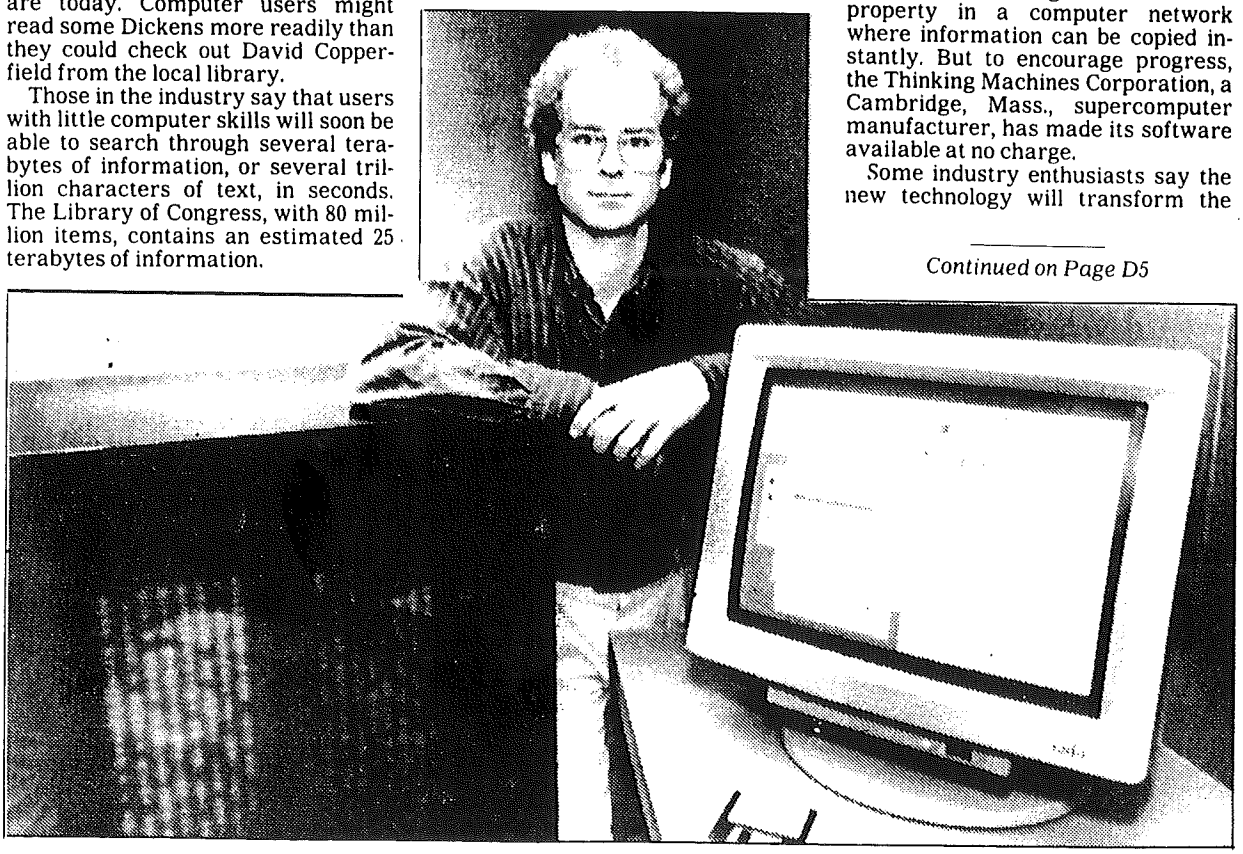
Such ready access to huge amounts of computerized information has been the dream of many in the industry. But a lack of computing power, effective software and high-speed digital networks has stalled progess until recently.

If many of the technical problems are being solved, major business and political disputes remain. The researchers acknowledge that they must resolve several questions of privacy and pricing before they can put the new methods to commercial use.

Many sources of information, like government documents, might be available free, but other services, including electronic newspapers, will be available only to those who pay. The industry has yet to settle on ways to protect and charge for intellectual property in a computer network where information can be copied instantly. But to encourage progress, the Thinking Machines Corporation, a Cambridge, Mass., supercomputer manufacturer, has made its software available at no charge.

Some industry enthusiasts say the new technology will transform the

Continued on Page D5



Mike Theiler for The New York Times

Brewster Kahle was the leader of the development team at the Thinking Machines Corporation for a nationwide computerized library system. His team's software links a CM2A Connection Machine, left, with a personal computer or work station like the Apple Macintosh II at right. Using high-speed data highways, the two machines can function together although they may be thousands of miles apart.

For Shakespeare, Just Log On

Continued From First Business Page

way computerized information is sold. Mitchell Kapor, the founder of the Lotus Development Corporation, predicts the growth of a new industry as significant as the personal computer business. Some companies, like Dow Jones & Company, that already provide computerized information over telephone lines have taken part in developing the new computer library.

The Search Is Simplified

In 1989, Thinking Machines enlisted the support of Dow Jones, Apple Computer Inc. and the KPMG Peat Marwick accounting and consulting firm to design the computer library, called Wide Area Information Servers, or WAIS (pronounced ways). The system permits computer users to quickly search through a huge volume of information even if it is stored at several distant locations.

The system lets users conduct searches by typing common English phrases instead of more complicated computer commands. While current systems like Dialog and Nexis require users to specify precisely the information they want, the new system can respond to a user's inferences. It initially presents a sample list of documents. The user chooses one or several, and then a "relevance feedback" program presents other documents most like the ones selected.

"This solves the problem of how to get to the information you need, getting not too much and not too little," said Esther Dyson, editor of Release 1.0, a computer industry newsletter.

> It will soon be possible to search through millions of items in seconds.

This is a sharp contrast to the way services operate today, Ms. Dyson said. A computer user may need to call seven or eight separate data bases depending on the kind of information needed.

The WAIS system lets users of Apple personal computers harness a network of Thinking Machines supercomputers and smaller "server" computers to search data bases stored by Dow Jones, KPMG and several corporations and universities. Users can also read electronic mail, enter their corporate electronic libraries and summon up a wide variety of documents, newspapers and magazines.

A 'Corporate Memory'

At Thinking Machines, the WAIS system serves as a "corporate memory," allowing employees to retrieve memos, documents and other internal information. Employees who may not be working together can share expertise.

"If someone did something in Los Angeles and I'm sitting in San Francisco, I may not know about the work," said Robin Palmer, a senior manager at Peat Marwick.

WAIS delivers information over the Internet, a collection of 2,600 high-speed public and private computer networks. This Government-sponsored system of data highways is rapidly being improved and turned to commercial uses.

The market for software that allows the rapid retrieval of computerized text is small but growing, according to industry analysts. In 1989, the United States had fewer than 60,000 users; by the next year, total sales were about $120 million. The Delphi Consulting Group expects the market to grow to 160,000 users and $235 million by 1992.

"Information retrieval technology is starting to spread from supercomputers all the way down to personal computers," said Brewster Kahle, a Thinking Machines scientist who has led the WAIS experiment.

The WAIS system is built on a procedure for retrieving information developed by librarians who initially set out to computerize their card catalogues. The procedure — known in the field as Z39.50 — now has the support of the Library of Congress, Apple, Sun Microsytems Inc., Next Inc., Dow Jones and Mead Data Central.

In the future, a special directory or "white pages" will keep an up-to-date list of all the separate sources on the network.

Apple has its own electronic library project, borrowing its name, Rosebud, from the movie "Citizen Kane." The three-year-old project is based on the WAIS system, but adds features including the ability for a user to develop a personalized electronic newspaper.

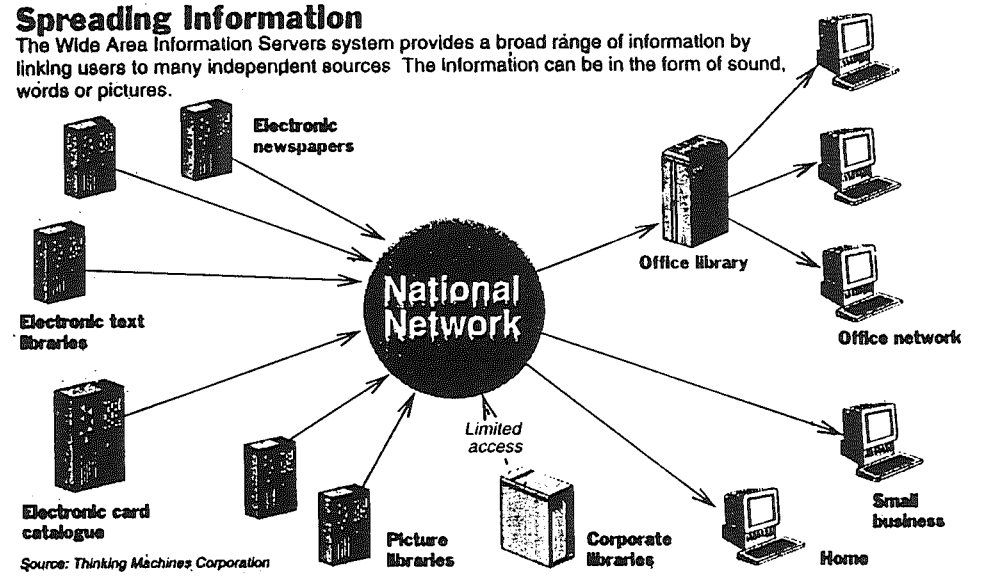
Rosebud uses special programs — called "reporters" — that let customers specify the kinds of information and news they want to retrieve from the WAIS system every day. Researchers at Apple's Advanced Technology Group said that in the future the necessary retrieval software might be a standard part of a computer's operating system.

They expect improvements in the Internet computer network to greatly lower the cost of information searches, promoting the introduction of many new services. The Government proposes to expand and improve Internet by financing a National Research and Education Network, or NREN, that could extend high-speed computer links into schools and communities across the country.

"With things like NREN, everything could change overnight," said Tim Oren, an Apple researcher.

Spreading Information

The Wide Area Information Servers system provides a broad range of information by linking users to many independent sources. The information can be in the form of sound, words or pictures.



Source: Thinking Machines Corporation

Wide Area Information Servers:
An Executive Information System for Unstructured Files

Brewster Kahle
245 First Street, Cambridge, MA 02142
Thinking Machines Corporation
brewster@think.com

Harry Morris
1010 El Camino Real, Suite 310, Menlo Park, CA 94025
Thinking Machines Corporation
morris@think.com

Franklin Davis
245 First Street, Cambridge, MA 02142
Thinking Machines Corporation
fad@think.com

Thomas Erickson
20525 Mariani Blvd., Cupertino, CA 95014
Apple Computer
thomas@apple.com

Clare Hart
5918 Red Coat Lane, West Bloomfield, MI 48322
Dow Jones & Co.

Robin Palmer
50 W. San Fernando Street, Suite 1200, San Jose, CA 95113
KPMG Peat Marwick
palmer.r@applelink.apple.com

November 1991

Abstract

In this paper we present a corporate information system for untrained users to search gigabytes of unformatted data using quasi-natural language and relevance feedback queries. The data can reside on distributed servers anywhere on a wide area network giving the users access to personal, corporate, and published information from a single interface. Effective queries can be turned into profiles, allowing the system to automatically alert the user when new data is available.

The system was tested by twenty executive users located in 6 cities. Our primary goal in building the system was to determine if the technology and infrastructure existed to make end-user searching of unstructured information profitable. We found that effective search and user interface technologies for end-users are available, but network technologies are still a limiting cost factor.

As a result of the experiment we are continuing the development of the system. This paper will describe the overall system architecture, the implemented subset, and the lessons learned.

1. Introduction

Systems that allow corporate executives to access personal, corporate, and published information such as memos, reports, manuals, and news are new in the field of information management. The first integrated systems are just now coming on the market. They exploit networking, online mass storage, and end-user search systems; each of these has existed for some time, but their combination and integration has not been available for the corporate environment.

Commercial systems exist in each of the personal, corporate, and published data areas with different levels of user friendliness. ON Location™, for instance, allows easy content based retrieval of personal files on a Macintosh, while Lotus Megellean™ performs a similar function on a PC. Verity's Topic™ system allows for searching of LAN-based (usually corporate) archives but primarily for a trained user community. Dialog, Dow Jones, and Mead Data are major online providers of published information, but again the majority of their users are professionals in the field of information retrieval (such as corporate librarians).

Academic systems have also been developed for some of these applications. The Information Lens project (Malone, 1986) revolves around structured electronic mail to help in automatic organization and retrieval of business information. Project Mercury (Ginther-Webster, 1990) is a remote library searching system that uses a client-server model. The Smart system (Salton, 1971) is an information retrieval system that embodies many different searching strategies. The SuperBook project (Egan, 1989) is working on user interfaces for information systems concentrating on the scientific user. Each of these systems is breaking new ground, but there is still no complete solution for the business executive wishing to search diverse information sources.

The Wide Area Information Servers (WAIS, pronounced "ways") system was constructed to test the acceptability of an integrated search system directly targeted at executives (Kahle, 1989). The companies participating in the project offered expertise in different parts of the problem: Dow Jones, with its business information sources; Thinking Machines, with its high-end information retrieval engines; Apple, with its user interface background; and KPMG Peat Marwick, with its information-hungry user base. Through this project, we wanted to find out if the wide area information retrieval market could incorporate users other

than those trained searchers who are familiar with a variety of query languages and databases.

In the WAIS project we used a general architecture and built a small implementation to test the feasibility of an integrated information retrieval system for corporate end users. This paper is a report on the overall architecture, the various implementations, and the lessons learned from this work.

2. The WAIS Architecture

The WAIS system took advantage of available technology to make a system which could then be tested on corporate executives to determine user acceptability. The system was composed of: clients, servers, and the protocol which connects them. The information servers were Connection Machine systems, running a parallel signature-based search algorithm (Stanfill, 1986). The cross-country network connected several LANs with leased lines running AppleTalk and TCP, and carrying a variation on the Z39.50 application protocol. The clients ran on Macintoshes. This section will describe the overall architecture, and the next section will describe what exactly was implemented and used during the experiment.

The WAIS architecture had the following goals:

- accessible to novice users - little or no training should be required in order to perform effective searches.
- remotely accessible - the servers must be accessible over a variety of networks.

- uniform interface - a variety of databases, whether personal, corporate or published, must be accessible from the same user interface.
- automatic alerting - it must be easy to create profiles for background searching
- scalable - the system must scale in number of servers, size of servers, and intelligence of servers.
- security - individuals and groups should be able to maintain control of who accesses their data.
- pricing model - a variety of information pricing structures, from per-minute charges, to subscriptions, must be supported.
- multimedia - the system must support the retrieval of any file format.

Many of these goals were achieved, while others, such as pricing model experimentation, were left unresolved.

In a client-server system, the client program is the user interface, the server does the searching and retrieval of documents based on indices, and the protocol is used to transmit the queries and responses. The client and server are isolated from each other through the protocol so that they can be physically distant and interchangeable. Any client which is capable of translating a user's request into the standard protocol can be used in the system. Likewise, any server capable of answering a request encoded in the protocol can be used. In order to promote the development of both clients and servers, the protocol specification is in the public domain, as is its initial implementation.

On the client side, searches are formulated as quasi-natural language questions. The client application then formats the query for the WAIS protocol, and transmits it over a network to a server. The server receives the transmission, translates the received packet into its own query language, and searches for documents satisfying the query. The ranked list of relevant documents are then encoded in the protocol, and transmitted back to the client. At this point, the servers do not "understand" the quasi-natural language question posed by the user in any sense that a human would, but they use the words and phrases in the question to find documents that use those terms. The client decodes the response, and displays the results. Documents of interest to the user can then be retrieved from the server.

2.1 Searching

We modeled the searching strategy on the interactive process people use when talking with a reference librarian. The library scenario is one where the patron approaches a librarian or researcher with a description of needed information. The librarian might ask a few background questions, and then draw from appropriate sources to provide an initial selection of articles, reports, and references. The patron sorts through this selection to find the most pertinent documents. With feedback from these trials, the researcher can refine the materials and even continue to supply the patron with a flow of information as it becomes available. Monitoring which articles were useful can help the researcher provide appropriate information in the future.

The WAIS system uses a similar means of interaction: the user states a question in unrestricted natural language to a set of sources, and a set of document descriptions is retrieved (see figure 1). The server assigns each document a score,

based on how closely the words in the document matched the question (see figure 2). The user can examine any of the documents, print them, or save them for future use (see figure 3). If the initial response is incomplete or somehow insufficient, the user can refine the question by stating it differently.

Once a good document is found, the user may say "I want more like this one" by marking the retrieved documents as being "relevant" to the question at hand, and then re-running the search (see figure 4). This method of query refinement is called "relevance feedback" (Salton, 1983). The server uses the marked documents to attempt to find others which are similar to them. In the present WAIS server, "similar" documents are ones which share a large number of statistically significant words and phrases. This brute force method works surprisingly well with large collections of documents (Stanfill 1986; 1991).

2.2 A Common Protocol for Information Retrieval

One of the most far-reaching aspects of this project was the development of an open protocol. The four companies involved jointly specified a standard protocol for information retrieval by extending an existing public standard, Z39.50-1988 (NISO, 1988). We choose this public standard rather than inventing one ourselves since it was close to what we needed and it could help us keep the protocol from being regarded as proprietary.

The use of an open and versatile protocol can foster hardware independence and competition. This not only provides for a much wider base of users, it allows the system to evolve over time as hardware technology progresses. For example, the protocol provides for the transmission of audio and video as well as text, even though at present most personal computers are unable to handle them. However, they are free to ignore pictures and sound returned in response to questions, and

to display and retrieve only text, if that is all they are capable of processing. Higher-end platforms are free to exploit their greater processing power and network bandwidth.

Z39.50 is a general attribute-based Boolean search protocol intended to run over the OSI stack. It was designed for search and retrieval of bibliographic (MARC) records in libraries. As such, its structure allows easy access to traditional Boolean search systems such as STAIRS (Salton, 1983).

The WAIS protocol is an extension of the existing Z39.50-1988 standard, but we are working with the standards committee to merge the extensions back into the newer versions (Davis, 1990). The extensions allow support for multi-media data, large documents, a directory of servers, different communication systems, and distributed retrieval. To support multi-media, a document must be available in a number of formats. This was accomplished by listing the set of available types in the search response from which the client can choose one to retrieve. Another problem with the protocol involved retrieving large records. Large documents, whether text or not, would be slow to display if the whole document had be retrieved at one time, as is required in the original standard. Large documents are supported in the WAIS protocol by allowing the client to retrieve sections of a document based on bytes or lines. We also standardized a format for describing servers (Kahle, 1991a) and how to contact them, which is necessary to implement a directory of servers. To support communication systems other than the full OSI protocol stack, a header was needed to show how long the packet was and how it was encoded. With this packet header we implemented the WAIS protocol over modems, TCP/IP, and X.25 systems. To support distributed retrieval we needed a document identifier system that could be used in a distributed environment (Kahle, 1991b).

The protocol used in the WAIS system has proven useful in the distributed full-text environments in which we tested it.

2.3 User Interfaces: Asking Questions

Users interact with the WAIS system through the Question interface. Each question form has an area for the user's quasi-natural language question, the list of sources which will be accessed to try to answer the query, the list of relevant documents, and a list of answer documents.

The illustrations here are taken from the initial WAIStation program produced at Thinking Machines for the Apple Macintosh. We have also built clients for X windows and gnu-emacs. Another Macintosh interface was developed that emphasizes the alerting feature (Erickson, 1991).

With most current retrieval systems, complications develop when one begins dealing with more than one source of information. For example, one contacts the first source, asks it for information on some topic, contacts the next source, asks it the same questions (most likely using a different query language, a different style of interface, a different system of billing), contacts the next source, and so on. One of the primary goals behind the development of the WAIS system was to replace all this with a single interface.

With WAIS, the user selects a set of sources to query for information, and then formulates a question. When the user presses the RUN button (see figure 2), the system automatically asks all the desired servers for the required information with no further interaction necessary by the user. Thus, the documents returned are sorted and consolidated in a single place, to be manipulated by the user. The user has transparent access to a multitude of local and remote databases.

From the user's point of view, a server is a source of information. It can be located anywhere: on the local machine, on a network, or on the other side of a modem. The user's workstation keeps track of a variety of information about each server. The public information about a server includes how to contact it, a description of the contents, and the cost. In addition, individual users maintain their own private information about the servers they use.

Users may need to budget the money they are willing to spend on information from particular servers, know how often and when each server is contacted, and assess the relative usefulness of each server. In the current interface, the budget entries were put in as placeholders, since all servers are currently free. When a source is contacted, all questions that refer to the source are updated with the new results.

A "confidence factor" allowed users to multiply the score returned from different servers so that the list presented to the user would be more appropriate. This was put in the interface to anticipate a number of different server technologies with different scoring algorithms. The "confidence factor" allows the user to adjust the scores. Also, a user might have a preference for the information from one server over another so a subjective balance would be helpful. This feature was rarely, if ever, used since the number of servers was small, they all used the same server technology, and most users only asked one source at a time anyway.

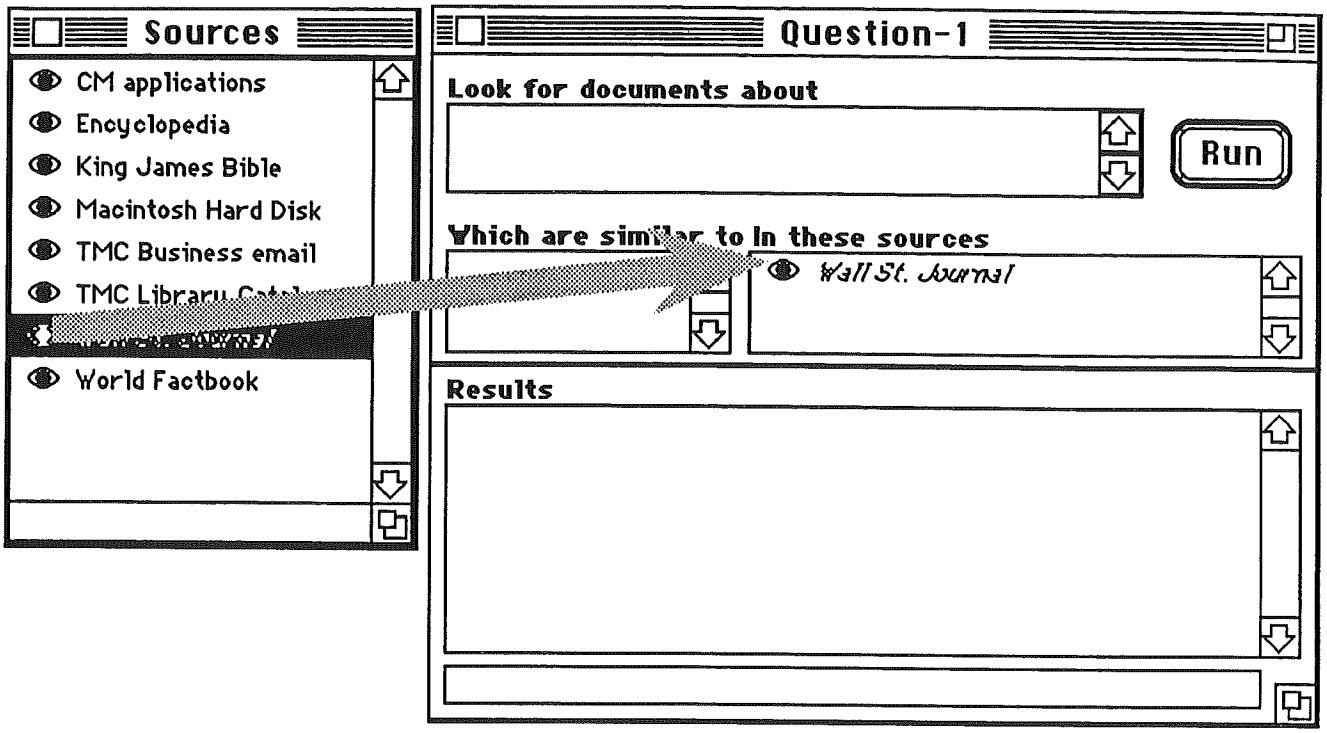


Figure 1: *Sources are dragged with the mouse into the Question Window. A question can contain multiple sources. When the question is run, it asks for information from each included source.*

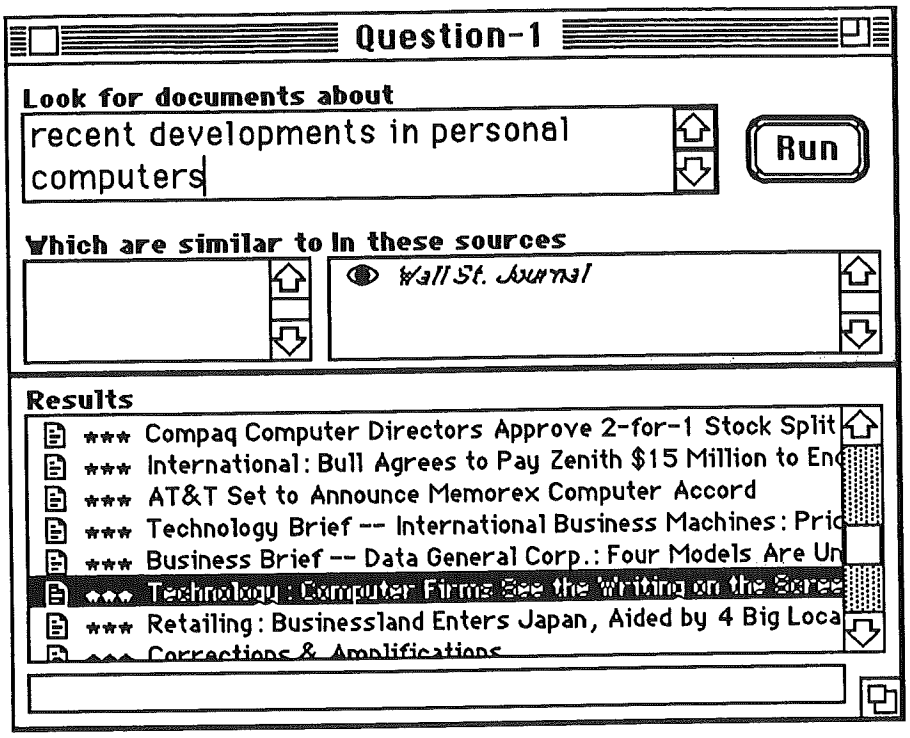


Figure 2: *When a query is run, headlines of documents matching the query are displayed.*

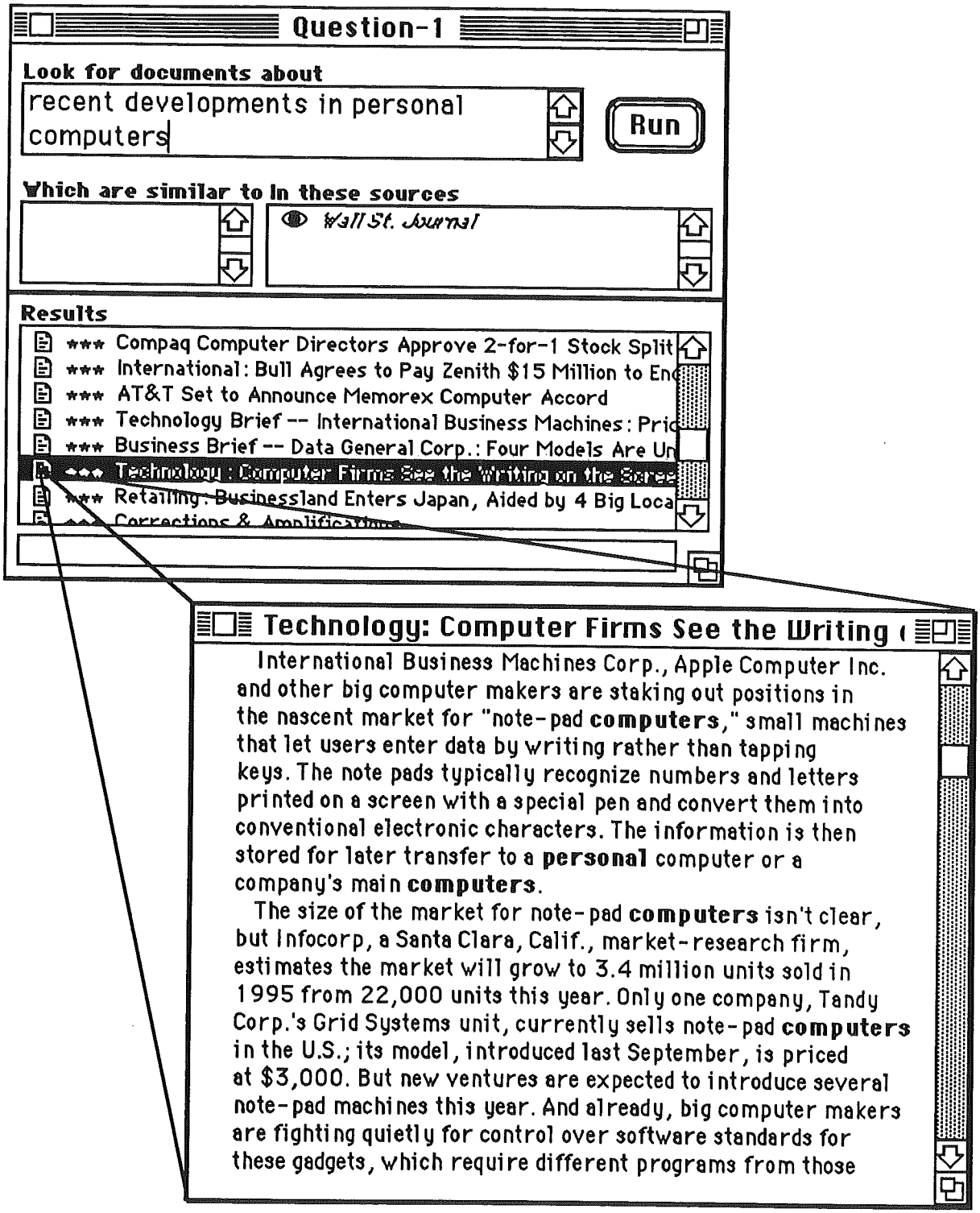


Figure 3: With the mouse, the user double clicks on any resulting document to retrieve it. The document can contain graphics.

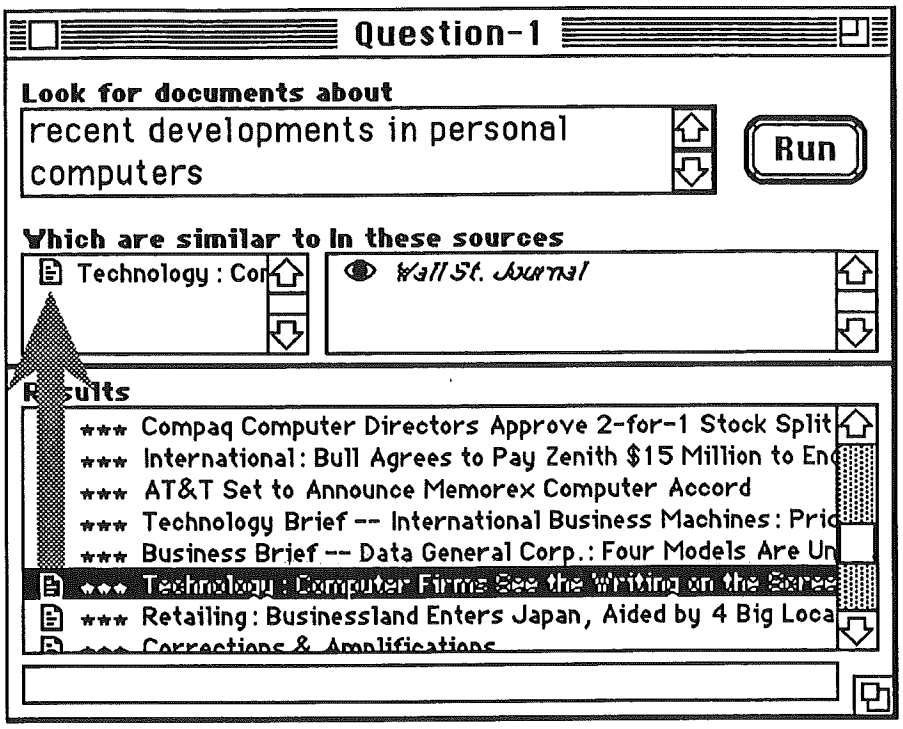


Figure 4: *To refine the search, one or more of the result documents can be moved to the "Which are similar to:" box. When the search is run again, the results will be updated to include documents which are "similar" to the ones selected.*

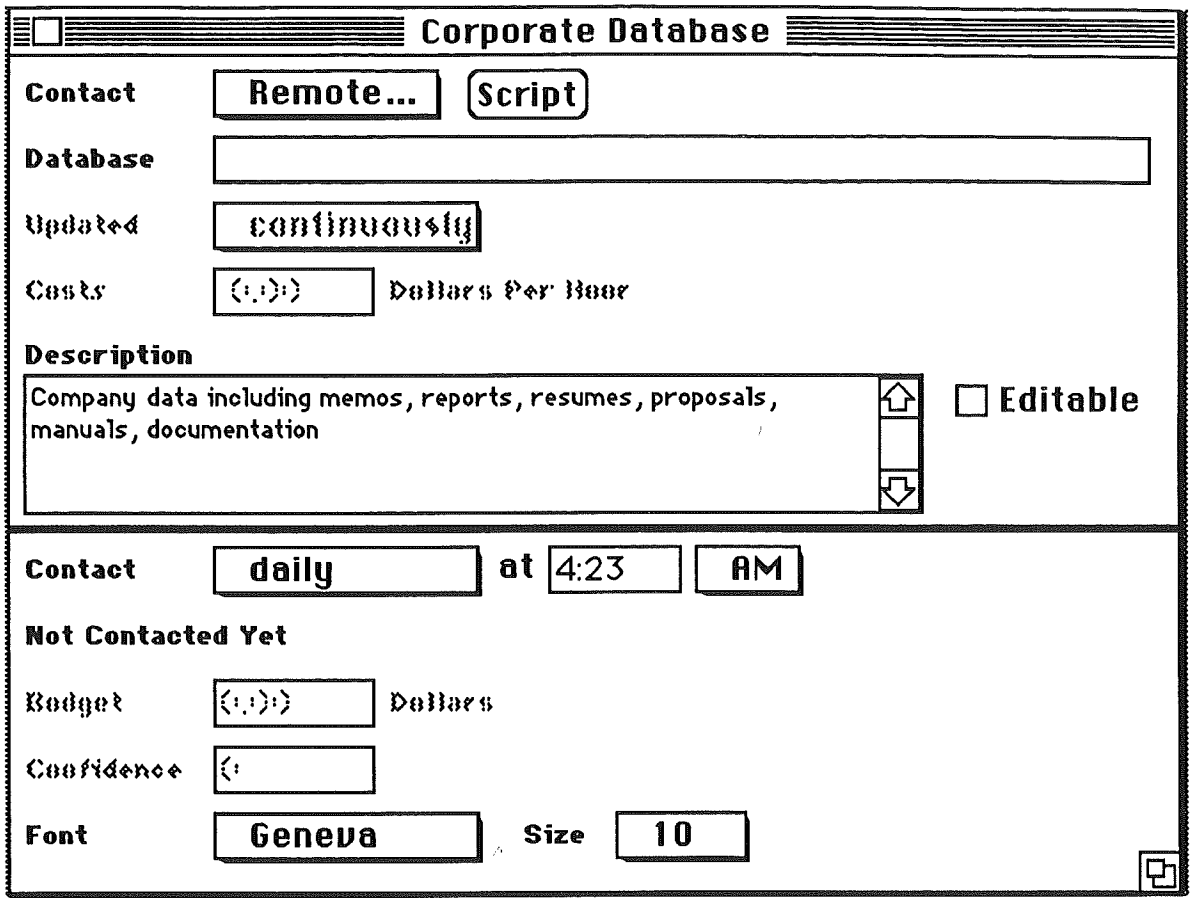


Figure 5: The Source description contains all the necessary information for contacting an information server.

2.4 Servers

The servers in the WAIS system hold databases that can be queried by a client. References of documents that best match the words and phrases in the query are returned to the client. A client can then request all or part of a document from the server. Since the client explicitly contacts the server, any number of billing methods could be employed such as 900 numbers, credit cards, and subscriptions.

The Connection Machine server system (CMDRS), used in the WAIS system, stores the documents in a compressed form, called signatures, which can be searched quickly using the parallel processors of the Connection Machine (Stanfill, 1986). The signatures are stored in the RAM of the machine thereby assigning a few documents to each processor of the machine. Each word in the query is then broadcast to all the processors, and a score is kept for each document to reflect the number of words and phrases that matched. Weighting is done based on crude proximity and occurrence frequency. The resulting search results have been found to be useful to end-users.

As the dissemination of information becomes easier, questions of ownership, copyright, and theft of data must be addressed. These issues confront the entire information processing field, and are particularly acute here. The WAIS system is designed to keep control of the data in the hands of the servers. A server can choose to whom and when the data should be given. Documents are distributed with an explicit copyright disposition in their internal format. This is not to say that theft cannot occur, but if a client starts to resell another's data, standard copyright laws can be invoked. By keeping the control of the distribution of works with the creators, many of the problems of copyright do not arise.

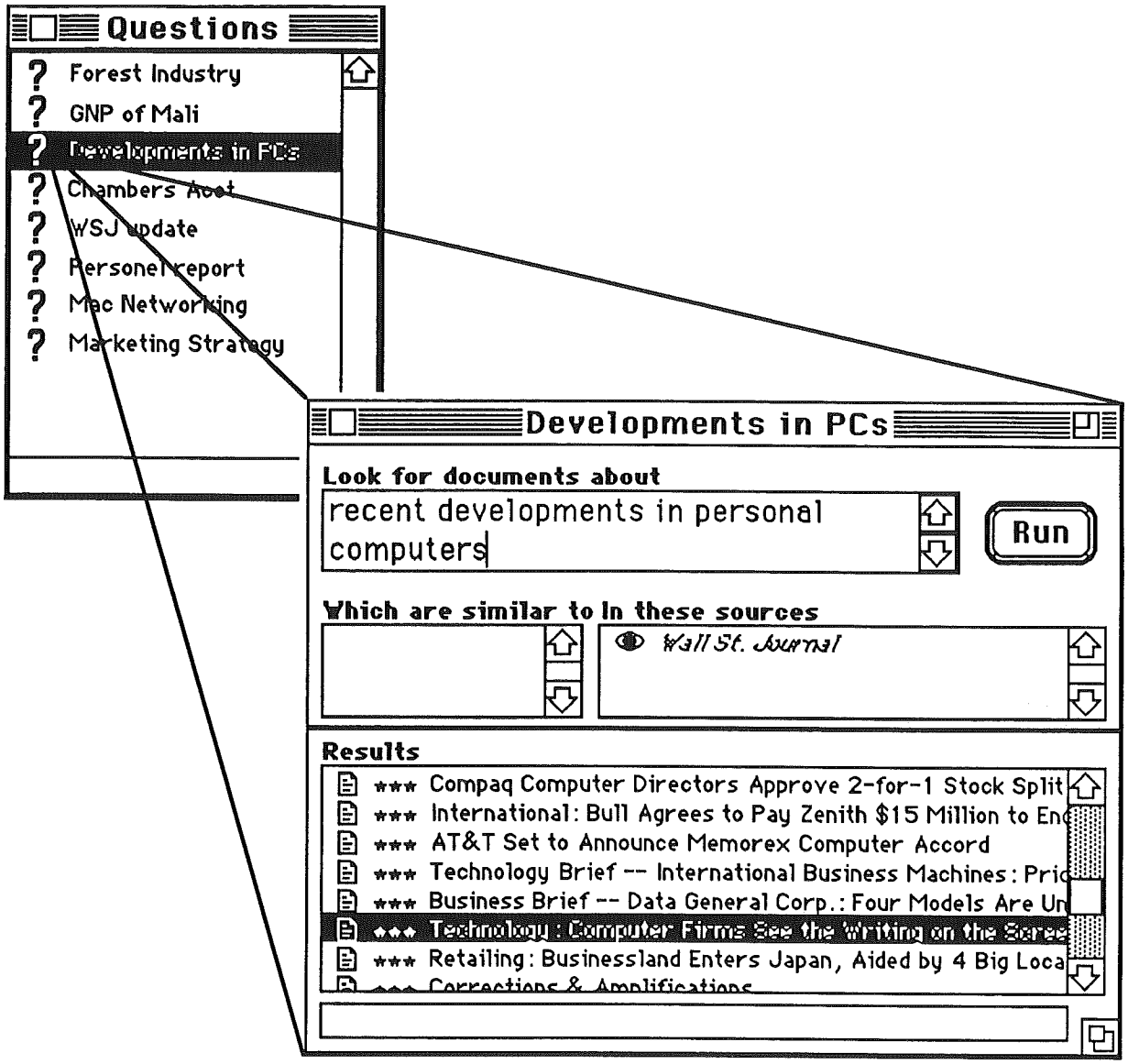


Figure 6: Opening a saved question which was automatically updated in the background, and contains new data.

2.5 Rerunning Questions - A Personal Newspaper

In addition to providing interactive access to information, the WAIS system can also be used as a rudimentary personal newspaper to alert its user when new documents are available on a subject that might be of interest (see figure 6). In the library literature, this is referred to as selective dissemination of information (SDI), and many manual, semi-automated, and automated systems have been implemented. Our initial implementation is to save interactive questions and automatically rerun them at periodic intervals checking if new documents were available. This technique has the advantage of hiding communication costs, using systems off hours, and finding potentially interesting information in a timely manner.

2.6 MultiMedia Database

The documents retrieved through WAIS may be any kind of file such as text, still graphics, motion pictures, or hypertext documents. The searching of the system is based on an initial quasi-natural language question and further relevance indications, but the server is free to use that information in any way to find appropriate documents. The protocol simply defines a document as a block of data and a type. The type is used by the client to determine how to display the document. A list of available types is part of the search response of each document. This allows clients to choose among a selection of types, and suppress documents whose types they can't display. Alternatively, they can simply store the documents in their local disk for latter processing. Our initial X windows clients are able to use other programs to display graphic data such as TIFF and GIF. The Macintosh client can display PICT images and text, but can theoretically download any type of file.

Non-textual data is indexed in one of two ways. If the data includes an embedded description (e.g. TIFF), the description is used for indexing. Otherwise an external description is indexed. When a search identifies the description file as a suitable response, the multi-media data is returned instead of the description file.

2.7 The Directory of Servers

To find sources of information in a distributed environment, we used a "directory of servers" which is a database of documents describing other servers. In response to a query, the database of servers is searched, returning a list of documents (i.e., server descriptions) which match the query. Instead of text documents, however, it takes advantage of the mixed type capabilities of WAIS to return a structured document with many specific fields for cost and contact information (see figure 5). This capability will become more important as the number of servers increases.

For example, suppose you needed information concerning the current gross national product of Mali, but had no idea on which server to find it. You could first ask the directory of servers for "information about the current economic condition of Mali." The directory will take the words in the query and find descriptions of the servers that contain those words. It might then return several documents. The World Factbook, for instance, might appear because of a match on "economic condition". This source description could then be used as the source field of another question. This time, the system would contact the World Factbook, ask for the information, and possibly return a document with a description of Mali (World Factbook, 1974).

Additionally, the directory of servers provides a means for information providers to advertise the availability of their data. When a new source becomes available,

the developers can submit a textual description, along with the necessary information for contacting the server. This information is added to the directory, and becomes available to the public by the searching interface.

3. The Prototype WAIS System

In the fall of 1990 we installed an experimental WAIS system at Peat Marwick. The prototype was used by 20 users in six cities. Peat Marwick utilized corporate data in Montvale, New Jersey, and Dow Jones information in Princeton, New Jersey. The system was run successfully for six months with good user reactions.

KPMG Peat Marwick is an example of an information-intensive company. Their role as consultants requires that they maintain an awareness of new products, market fluctuations, changing laws, internal regulations, and competition. In addition, as a large organization, there is considerable internal information that can be leveraged, such as company contacts, bids, reports, and resumes. Furthermore, their distribution in 40 countries, with 200 offices in the Uunited States alone, makes them a prime candidate for wide area information technology.

The primary users were located in San Jose and connected by 56kbaud and 9.6kbaud circuits to the servers in New Jersey. The 20 managers and partners in the Peat Marwick's accounting division used an 8192 processor Connection Machine system for serving reports, proposals, resumes, contracts, accounting manuals, the Peat Marwick Audit Manual, Management Guide, and Professional Development Courses, documents from the Financial Accounting Standards Board, the Government Accounting Standards Board, and the American Institute of Certified Public Accountants, and a tax library. The data

were separated into twelve different databases, which could be searched separately or in any combination. There was also a virtual database consisting of all of these sources.

The connection to Dow Jones provided access to 1 Gigabyte of data, running on a 32K processor Connection Machine. The data consisted of a year of the Wall Street Journal, Barrons, and 400 magazines. Each of the approximately 250,000 articles was a separate document. The ability to search personal data was not available at the time of the experiment.

3.1 Lessons Learned

The search technology performed well in finding useful data for end users who were given little instruction about system use. The speed of the searches (usually between two and ten seconds) depended on the communication speed, since the search itself took much less than a second. When the response time was greater than 10 seconds, the users voiced complaints, but in general they were very pleased with the search results. The ability to execute searches without prior training, and without in-depth knowledge of the database was essential to the users. Relevance feedback was used frequently and effectively by users who were aware of its existence. Not all users realized it was available, however. This is an opportunity for improving user interfaces. For example, relevance feedback could be performed automatically on any document which the user chooses to view. This would result in a kind of automatic, dynamically linked hypertext system, where every document is "linked" to all similar documents.

The Macintosh user interface (WAIStation) also performed well in terms of ease of use and adaptability. With a single demonstration, most users were able to execute searches and save their results. Left with only the manual, new users

took 15 to 30 minutes to feel comfortable with the system. The ability to transparently search local and remote databases was greatly appreciated, as reported in user feedback forms. The biggest problem we had with the interface was in implementing the TCP and modem connections from the Macintosh. The automatic updating feature of WAIStation was rarely used and needs more work to make it more obvious and to allow it to give better feedback when documents are found.

Wide area communications proved to be a difficult part of the project due to our resistance, based on future cost projections, to use leased lines. The original plan called for linking San Jose and Montvale with Shiva Telebridges™ running at 9600 baud on a normal phone line. This approach did not prove reliable, nor did it give us reasonable performance. We ended up replacing this link with a dedicated 56kbaud line attached to a SyncRouter (Engage Communications™). The dedicated line was highly reliable and 56kbaud was fast enough to support many active users of the system, while maintaining an interactive feel in both search and retrieval.

Organizing and formatting the data for display on the client workstation proved to require more effort than we expected. The current Macintosh client is capable of displaying only ascii text and PICT format picture files. This meant that the corporate data, which consisted primarily of word processor files, had to be converted to ascii. Since the conversion was not perfect, some documents required a small amount of manual reformatting. This is obviously unacceptable in a production system. A more attractive solution might be to build a client which can display the most common document formats, and which can call on other applications to display formats it doesn't understand. This approach will become easier to implement as document filters (e.g., Claris™ XTND) and

interprocess communication become more common. This approach will also make it possible to index and store the original document, rather than an ascii shadow.

As the searchable Peat Marwick corporate collection grew, the users wanted to search just parts of the database. The natural divisions for the users were the original sources of the text, such as training manuals or government legal texts.

In summary, we found that the users were pleased with the system, and some used it many times each day. It appears that there is a market for end user search systems, and that the technology is ready. The weak link seems to be communication infrastructure.

4. Conclusion

In developing the WAIS system, the participating companies have demonstrated that current hardware technology can be used effectively to provide sophisticated information retrieval services to novice end users. How this might affect information providers is not yet understood. The users at Peat Marwick found the technology useful for day-to-day tasks such as researching potential new accounts and finding resources within their own organization. Since these tasks are not restricted to the accounting and management consulting industries, we are optimistic that this type of technology can be fruitful and productive in many corporate settings.

The future of this system, and others like it, depends upon finding appropriate niches in the electronic publishing domain. Potential uses include making current online services more easily accessible to end-users, and allowing large corporations to access their own internal data more effectively. It is also possible

that near-term development will focus on a single professional field such as patent law or medical research.

Acknowledgments

The design and development of the WAIS Project has been a collective effort, with contributions and ideas coming from many people. Among them:

Apple Computer: Charlie Bedard, David Casseras, Steve Cisler, Ruth Ridder, Eric Roth, John Thompson-Rohrlich, Kevin Tiene, Gitta Soloman, Oliver Steele, Janet Vratny-Watts.

Dow Jones News/Retrieval: Rod Wang, Roland Laird.

KPMG Peat Marwick: Chris Arbogast, Mark Malone, Tom McDonough.

Thinking Machines: Dan Aronson, Patrick Bray, Jonathan Goldman, Danny Hillis, Rob Jones, Barbara Lincoln, Gordon Linoff, Chris Madsen, Gary Rancourt, Sandy Raymond, Steve Schwartz, Tracy Shen, Craig Stanfill, Robert Thau, Ephraim Vishniac, David Waltz, Uri Wilensky.

References

(Davis, 1990) F. Davis, B. Kahle, H. Morris, J. Salem, T. Shen, R. Wang, J. Sui, and M. Grinbaum, "WAIS Interface Protocol Prototype Functional Specification," Thinking Machines, April 1990. Available via anonymous ftp: /pub/wais/doc/wais-concepts.txt@quake.think.com or wais server wais-docs.src.

(Egan, 1989) D. Egan, J. Remde, L. Gomez, T. Landauer, J. Eberhardt, and C. Lochbaum, "Formative Design-Evaluation of SuperBook," ACM Transactions on Office Information Systems, January 1989.

(Erickson, 1991) T. Erickson and G. Salomon, "Designing a Desktop Information System: Observations and Issues," Proceedings of the ACM Human Computer Interaction Conference, 1991. ACM Press, April 1991, pp. 49-54.

(Ginther-Webster, 1990) K. Ginther-Webster, "Project Mercury," AI Magazine, 1990, pp. 25-26.

(Kahle, 1989) B. Kahle, "Wide Area Information Servers Concepts," Thinking Machines technical report TMC-202, November 1989. Available via anonymous ftp: /pub/wais/doc/wais-concepts.txt@quake.think.com or wais server wais-docs.src.

(Kahle, 1991a) B. Kahle and H. Morris, "Source Description Structures," February 1991. Available via anonymous ftp: /pub/wais/doc/source.txt@quake.think.com.

(Kahle, 1991b) B. Kahle and H. Morris, "Document Identifiers or International Standard Book Numbers for the Electronic Age," May 1991. Available via anonymous ftp: /pub/wais/doc/doc-ids.txt@quake.think.com.

(Malone, 1986) T. Malone, K. Grant, and F. Turback, "The Information Lens: An Intelligent System for Information Sharing in Organizations; In Human Factors In Computing Systems," CHI'86 Conference Proceedings (Boston, MA); ACM: New York, 1986; pp 1-8.

(NISO, 1988) "Z39.50-1988: Information Retrieval Service Definition and Protocol Specification for Library Applications," National Information Standards Organization (Z39), P.O. Box 1056, Bethesda, MD 20817. (301) 975-2814. Available from Document Center, Belmont, CA. Telephone 415-591-7600.

(Salton, 1971) G. Salton, "The Smart Retrieval System - Experiments in Automatic Document Processing," Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

(Salton, 1983) G. Salton and M. McGill, "Introduction to Modern Information Retrieval," McGraw-Hill, New York, 1983.

(Stanfill, 1986) C. Stanfill and B. Kahle, "Parallel Free-Text Search on the Connection Machine System," Communications of the ACM 29,12, December 1986, pp.1229-1239.

(Stanfill, 1991) C. Stanfill, "Massively Parallel Information Retrieval for Wide Area Information Servers," International Conference on Systems, Man, and Cybernetics, Charlottesville, Virginia, October 1991.

(World Factbook, 1974) Publishing Sciences Group, Acton, Massachusetts, 1974.

Interfaces for Wide Area Information Servers

1/17/92 Version 0.8

Brewster Kahle, Harry Morris, Jonathan Goldman
Thinking Machines Corporation

Thomas Erickson
Apple Computer

John Curran
NSF Network Service Center

Email addresses: Brewster@think.com, morris@think.com, jonathan@think.com, thomas@apple.com, jcurran@nnsc.nsf.net

ABSTRACT

Interfaces for information access and retrieval are a long way from the ideal of the electronic book that you can cuddle up with in bed. Nevertheless, today's interfaces are coming closer to supporting browsing, selection, and retrieval of remote information by non-technical users.

The Wide Area Information Server (WAIS) system is built on a standard protocol, and thus supports access to a wide range of remote databases. As a consequence, it is a good platform on which to explore the design of interfaces for information retrieval. This paper describes 5 interfaces to WAIS that have been designed and implemented: WAIStation for the Macintosh, Rosebud for the Macintosh, XWAIS for X Windows, GWAIS for Gnu-Emacs, and SWAIS for dumb terminals.

The interfaces to WAIS described here reflect a variety of design constraints. Such constraints range from the mundane—coping with dumb terminals and limited screen space—to the challenging. Among the challenges addressed are how to provide passive alerts, how to make information easily scannable, and how to support retrieval and browsing by non-technical users. There are a variety of other issues which have received little or no attention, including budgeting money for access to 'for pay' databases, privacy, and how to assist users in finding out which of a large (changing) set of databases holds relevant information. We hope that the challenges we have identified, as well as the existence and public availability of source code for the WAIS system, will serve as a stimulus for further design work on interfaces for information retrieval.

1. INTRODUCTION

It requires little prescience to predict that one day computers will put an ocean of information at the finger tips of a vast population of users. However, although there is a considerable amount of information available from remote sources, the bulk of it is accessible only to information professionals, or users with highly technical backgrounds. A variety of obstacles effectively block the ordinary user from accessing information via the computer. These obstacles include the difficulty of locating appropriate information sources, the cumbersome maneuvers needed to get on-line and to connect to remote sources, and cryptic query languages. Furthermore, even if a user has succeeded in accessing a remote information source, it is likely that it will have its own special purpose interface, which may or may not support the user's needs.

In this paper we describe the Wide Area Information Servers (WAIS) project, which provides a protocol-based mechanism for accessing a variety of remote, full text information servers. WAIS has the potential for supporting a single interface to a wide variety of information sources, and offers a good platform on which to explore the design of interfaces for information retrieval. After a summary of existing information retrieval systems, we describe the WAIS system, and then describe the 5 interfaces which have been designed for it. In the course of these descriptions we discuss design constraints, interface issues, and practical matters which impacted the designs. We conclude with a summary, and some remarks on important issues which have not been addressed, and a invitation for other investigators to use the WAIS system as a platform for exploring interfaces to multiple, remote information sources.

2. BACKGROUND

2.1 Existing Systems

While a review of all existing systems is beyond the scope of this paper, it is useful to list a number of the most popular or significant interfaces for information retrieval.

Commercial interfaces for accessing full text resources on computers can be broken down into dialup services, local file access, and LAN-based access tools. Dialup systems such as Dialog and Dow Jones offer TTY interfaces to users, with menus and command lines being the dominant access tools. Some dialup services are offering client programs that run on personal computers to add graphical interfaces such as "Navigator" by Compuserve. In general, these interfaces are unique to the information provider. Local file access through full-text indexing has been achieved in command line form (e.g. the unix command "grep") and in screen based interfaces (e.g. ON Location (ON), and Digital Librarian (NeXT)). These interfaces often give browsing and searching capabilities for local files. Some of these interfaces have been stretched to work with files on file servers. LAN-based access tools usually use some sort of query language to access servers on the net, such as Verity's Topic system (VERITY), and numerous library systems. These query languages require some user training. Integrated tools for cross platform, cross vendor information access are not currently available in other systems.

A variety of research projects have explored information retrieval systems. The SuperBook project (Egan, 1989) targets users of static information. Project Mercury (Ginther-Webster, 1990) is a remote library searching system that uses a client-server model. Information Lens (Malone, 1986) is a structured email system for assisting in managing corporate information. NetLib for software (Dongarra, 1987) and Mosis for information on how to fabricate chips (Mosis) are examples of email based information retrieval systems.

2.2 Overview of the WAIS Project

The ultimate goal of WAIS was to define an open protocol which would allow any user interface or information server that talked the protocol to interact with any other component which used the protocol. From the user's perspective, this would mean that user interfaces and information sources could be mixed and matched, according to the user's needs.

WAIS started as a joint project between Thinking Machines Corporation, Apple Computer, Dow Jones & Co., and KPMG Peat Marwick (Kahle, 1991a). The proximate goal was to define the open protocol and demonstrate its feasibility by implementing and demonstrating a multi-vendor system which provided ordinary users with access to a variety of remote databases. Thinking Machines contributed its Connection Machine based retrieval technology,

Apple contributed its user interface expertise, and Dow Jones & Co. provided access to its commercial information sources. KPMG Peat Marwick provided access to its corporate data, and served as a test site. The resulting system was installed at KPMG Peat Marwick and enabled the designers to study the success of the system in a real world context. The existing system uses pseudo natural language queries, relevance feedback to refine queries, and accesses full text, unstructured information sources. These technologies were used because they had already been tested independently, thereby leading to faster implementation of the complete system.

After the collaborative phase of the WAIS project came the Internet experiment. In this phase of WAIS, source code for the open protocol, information servers, and for several interfaces were made freely available over the internet. In addition, Thinking Machines established and maintained a directory of information servers, which WAIS users could query to find out about available information sources. This phase of WAIS is still in progress, and has resulted in the creation of new interfaces, the availability over the internet of more than a hundred servers on three continents, and over 100,000 searches of the directory of servers. In the first 6 months of the Internet experiment, approximately 4000 users from 20 countries have tried this system, with no training other than documentation (Kahle, 1991b). Administrators of popular information servers indicate that they are getting over 50 accesses a day from many countries.

2.3 The WAIS System

WAIS employs a client-server model using a standard protocol (based on Z39.50) to allow users to find and retrieve information from a large number of servers. The client program is the user interface, the server does the indexing and retrieval of documents, and the protocol is used to transmit the queries and responses. Any client which is capable of translating a user's request into the standard protocol can be used in the system. Likewise, any server capable of answering a request encoded in the protocol can be used.

A WAIS server can be located anywhere that one's workstation has access to: on the local machine, on a network, or on the other end of a modem. The user's workstation keeps track of a variety of information about each server. The public information about a server includes how to contact it, a description of the contents, and the access cost.

The WAIS protocol (Davis, 1990) is an extension of the existing Z39.50 standard (NISO, 1988) from NISO. It has been augmented where necessary to incorporate many of the needs of a full-text information retrieval system. To allow future flexibility, the standard does not restrict the query language or the data format of the information to be retrieved. Nonetheless, a query convention has been established for the existing servers and clients. The resulting WAIS Protocol is general enough to be implemented on a variety of communications systems.

The WAIS clients will be described in detail in the next several sections. However, all of them work in a basically similar way. On the client side, queries are expressed as strings of words, often pseudo natural language questions. The client application then packages the query in the WAIS protocol, and transmits it over a network to one or more servers. The servers receive the transmission, translate the received packet into their own query languages, and search for documents satisfying the query. The lists of relevant documents are then encoded in the protocol, and transmitted back to the client. The client decodes the response, and displays the results. The documents can then be retrieved from the server. The documents can be in any format that the client can display such as word processor files or pictures.

3. WAISTATION: AN INTERACTIVE QUERY INTERFACE

WAIStation At A Glance	
Target Machine	Macintosh Plus and above, 9" Monochrome screen.
Effort	1 man-year
Number of Users	2000
Status	finished, freely distributed
Language	ThinkC
Communications	TCP/IP and Modem (not supported)
Designer	Harry Morris
Organization	Thinking Machines
Availability	Available for anonymous FTP from /public/wais/WAIStation*.sit.hqx@think.com
Design goals	Implementable quickly, support interactive queries well, changeable based on user's comments, make something very simple to learn (partner friendly), try out many ideas: interactive queries, passive alerting, asking multiple servers.
Used	In a study with accountants and tax consultants at KPMG: very good user acceptance. In the Internet experiment: estimated that half of the uses of WAIS are using WAIStation. (based on when the directory of servers did not work for Macintoshes, usage dropped to half).
Problems	dealing with the directory of servers (s). Modem code was difficult to get right.

WAIStation was designed for use in the WAIS experiment at KPMG Peat Marwick. As such, we needed an interface that would be easy to use, and would encourage successful searches by users untrained in search techniques. Peat Marwick often sends its employees into the field toting their Macintosh SE's along for use as portable computers. Thus we had to design the interface to run on a 9-inch black-and-white screen, and make minimal demands on CPU and memory. Furthermore, WAIStation was designed for use over modems and slow LANs.

3.1 Design Rationale

In designing WAIStation, we were informed by two metaphors - search as conversation, and storage by file folder. The process of formulating an effective search is highly interactive. Of the documents which match a query, the ones which match "best" are displayed. One or more may be of interest, in which case, they can be fed back to the system, interactively improving the search. We choose to view this process as a conversation. Thus the initial natural language question becomes that starting point for give and take between the user and the server(s). Relevance feedback provides the context for the question. As the search proceeds, some results may suggest alternative searches or branches of the conversation. This is provided for by allowing several questions to evolve at the same time.

Eventually one or more questions may be refined to the point where they are finding consistently good results. At this point, the question can be automated, becoming a dynamically updated file folder. At intervals these questions wake up and query their servers. The results are stored in the results field for later inspection. They can be thought of as regular Macintosh folders, except augmented with a charter describing how to keep their contents up to date.

This parallel with the Macintosh folder structure suggested a drag and drop construction for the user interface itself. Constructing a question is a three step process - typing the key words, specifying the servers to use, and specifying the relevant documents to feed back. If we think of questions like Macintosh folders, we can use the Macintosh's drag and drop mechanism for putting sources and relevant documents into a question. This approach makes WAIStation's mechanics instantly familiar to users of the Macintosh finder.

3.2 Human Interface

When WAIStation starts up, two windows appear – one contains the users available Sources (see below) and one contains the users saved Questions. Sources are identified by an eye icon, questions by a question mark icon.

Double clicking on a question icon opens the stored question, including any new results found since the last time it was examined. The top half of the question window contains a field in which to type key words (the natural language part of the question), a list of relevant documents, a list of sources, and a list of result headlines. Sources can be added to the question buy selecting a source icon (in the Sources window), and dragging it into the question. Relevant documents are specified in the same way.

Result documents, returned by the servers, can be examined by double clicking on their icon. Note that the result list contains a graphical indication of how well each document matches the query. The original graphic was a series of 0 to 4 stars, similar to the ratings found in TV guide. We thought that this rating scheme would be easily recognized. Experience proved that the stars did not provide enough information to be recognized, or to discriminate among the documents. Latter versions of the software replaced the stars with a horizontal bar giving 20 levels of resolution.

Any of the resulting documents can be opened and viewed in its own window. WAIStation supports plain ascii documents as well as PICT format pictures. Text windows automatically scroll to the position which the server considers the most relevant part of the document. This allows the user to quickly determine if a file is useful. In order to perform well over slow communications channels (modems and slow LANs) the text is downloaded on demand in 15 line chunks. The keywords used in the query are automatically highlighted in bold.

Sources are specially formatted text files which describe information servers and how to get to them. Double clicking on a source displays a window with several controls. The top part is information specified by the server itself - a pop-up menu to specify the method of contacting the server (ip-address/tcp-port, modem number and speed, or location of a local index); a script to run after logging in (for use by modems); a database to search (servers can support multiple databases); a display of when the server is updated, how much it costs to search, and a textual description of the databases' contents. The bottom half of the source window allows the user to specify personal information about the server - when to contact it (for automatic update); when it was last contacted; how much to spend on it; how much credence its results should be given (this is used to scale document scores, which helps in the sorting of responses to questions asked of multiple servers); the number of documents to ask for when searching it; and finally the font and type size to use when displaying plain text results (important to publishers). Several of these fields are merely place holders in the current implementation. In particular, budget and confidence have not been implemented yet since there are no for-pay servers yet, and the number of sources is still relatively small.

Source files can also be retrieved from servers. This allows users to search servers whose database elements are pointers to other servers. The results can be used as targets for further searches. An experimental directory of servers is being maintained on the Internet.

3.3 Implementation

WAIStation was implemented in Think C 4.0 using the object oriented class library. It took about a man year of effort. The most difficult parts were the automatic update facility and the communications. Automatic Update required the ability to do background processing - which is not a normal part of the Macintosh operating system. Communications were difficult primarily because we were simultaneously debugging the Z39.50 protocol, modem code, and the (then new) Apple Communications Toolbox. We eventually left modems unsupported, and replaced the Communications Toolbox with direct calls to MacTCP. Through this experience we found that communications speeds of less than 9600 baud were barely tolerable for interactive text retrieval.

3.4 Observations

We estimate that WAIStation is now in use by over 2000 users in twenty countries. The common user complaints center around configuring MacTCP, using (the undocumented) directory-of-servers, and avoiding a bug requiring the software to be installed on the start up disk.

We have noticed several shortcomings in the current design:

- Users want access to their own data - WAIStation is capable of searching a Macintosh based inverted index file, but we unbundled the index builder when we realized how much work it would take to make it useful under Macintosh OS. OnLocation (On Technology) is an implementation of a Macintosh indexer that could be used.
- Interaction with the directory of servers is incomplete - It is not obvious which search results are source files, and what to do with the ones that are. It should be possible to drag a retrieved source directly into a question's source window, but the present interface requires that it be saved first. The lesson we learned was that special cases should be handled specially, rather than forcing users to use general techniques "for consistency's sake".
- Printing documents and searching for keywords in documents (find/find-next) are simple functions which users expect.
- People want to see their documents in their original form - WAIStation currently only displays ascii and PICT. This can be fixed with format filters such as Claris' XTND, at the expense of the ability to download arbitrary sections of a document, since such filters require that the document be processed from the beginning.
- Relevance feedback was not obvious - users unfamiliar with the use of relevance feedback did not think to use it - it needs to be made more automatic. One way to do this might be to extend the notion that a question is a conversation, with relevance feedback as context (or body language) - clients or servers can be written that watch their users, and deduce which documents were relevant based on which ones were read. A simpler approach might be to always do relevance feedback, presenting the results in a "see also" list. We tried this, but the Macintosh was too slow to make it useful.
- Communications over 2400 baud modems are too slow to support interactive queries. We found that 9600 baud is barely acceptable, while 56Kb is sufficient to support several users.
- The finder-like interface (drag and drop) is not obvious - Even though the Macintosh Finder is based on drag and drop, no one expected it in an application. Once users were shown what to do, it was very natural. It was also not necessarily the best use of screen space, since it required that both the start and end of the drag be visible on the screen at the same time. Another anomaly worth mentioning is the fact that although we were simulating the finder, we had no "trash can" analogy. Removing a source was accomplished by dragging it onto the desk top and dropping it there, which confused some users.
- The alerting system was crude. For example, there was no visual cue to tell the user that a question had found new documents in the background. Also, the background searches did not exclude previously read documents.
- Headlines often don't give enough context - The headlines displayed in the question window were only about 60 characters long, making it difficult to identify which documents were useful without opening them. Furthermore, there was no provision to display the document's date or the name of the source it came from.

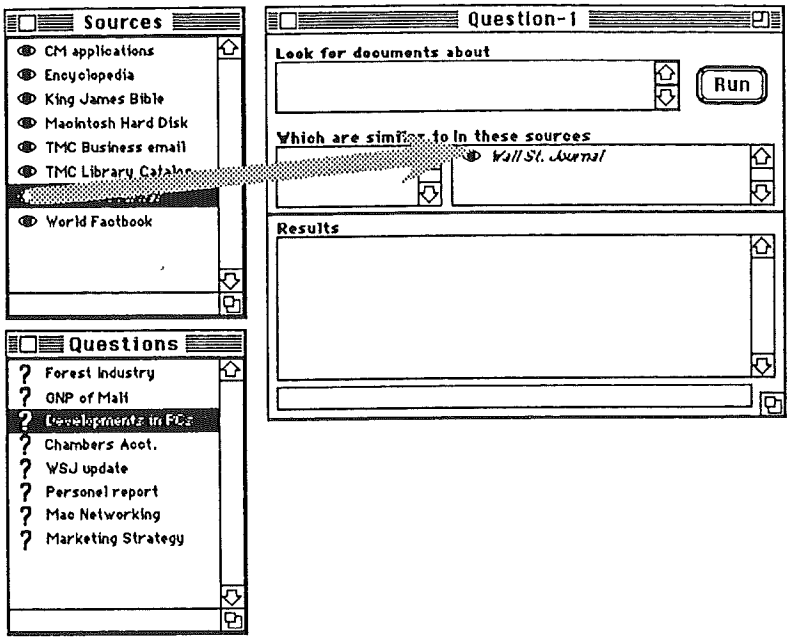


Figure 1 *WAIStation's Sources and Questions windows store the user's personal objects. Dragging a source into a question window specifies that the question will contact the source in order to fulfill its charter.*

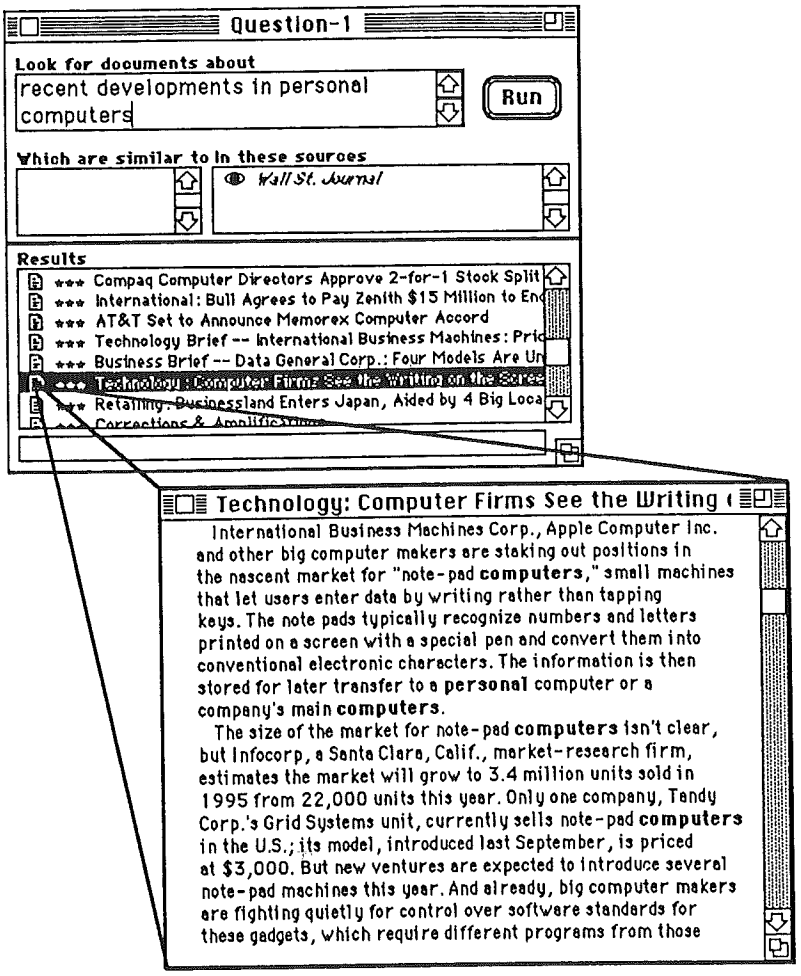


Figure 2 *After running the question, results are displayed in a scrolling list. Double clicking on a result opens a document window. Query words are highlighted.*

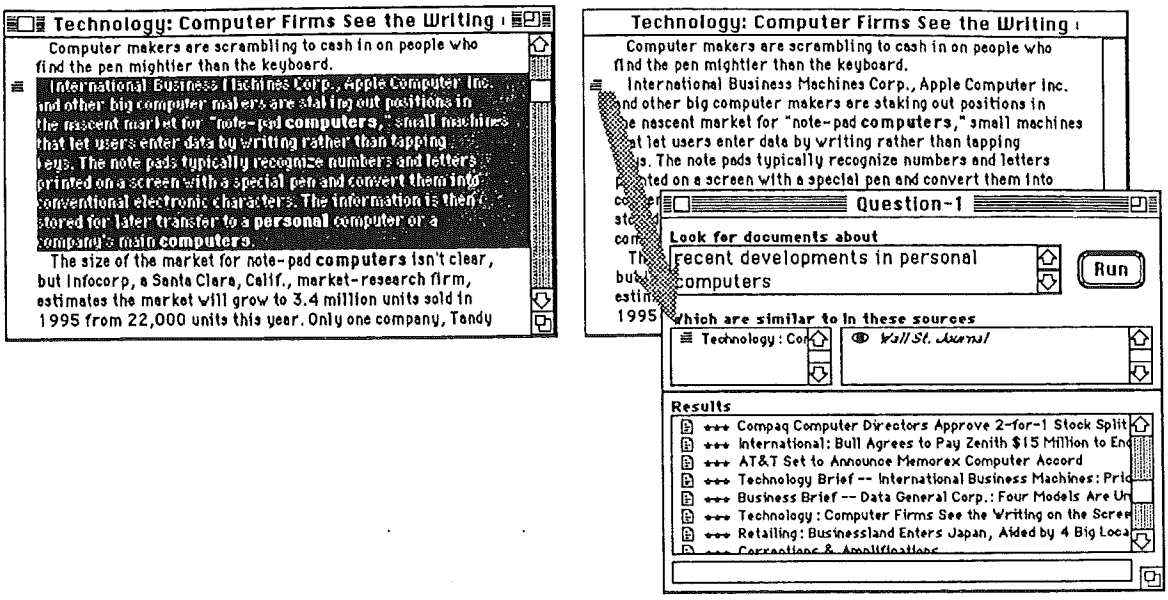


Figure 3 *Relevance feedback is done by selecting a document or part of a document, and dragging the document or paragraph icon into a question.*

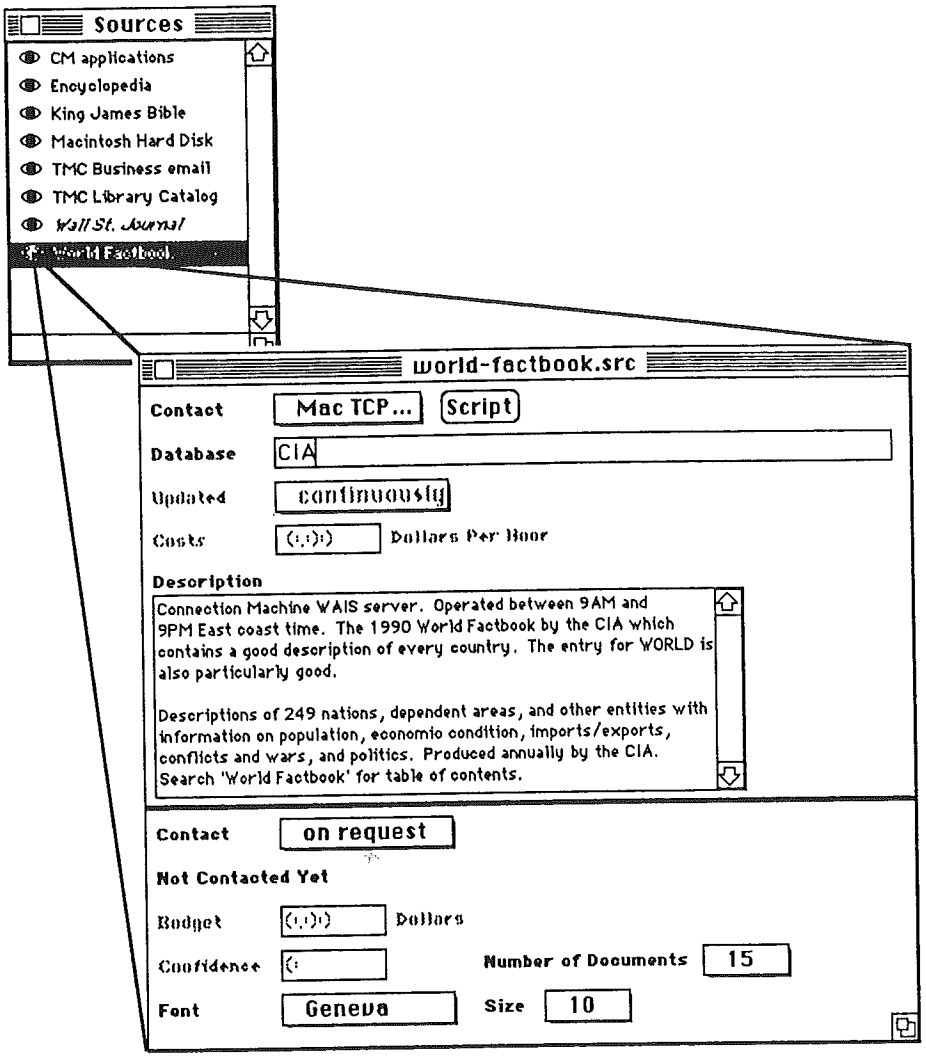


Figure 4 *Double clicking on a source icon opens a source window.*

4. THE ROSEBUD INTERFACE: REPORTERS AND NEWSPAPERS ON THE MACINTOSH

ROSEBUD At A Glance	
Target Machine	Macintosh II, color screen
Effort	5 man-years: 1 man-year HI designer with psychology background, 6 man-months HI designer with engineering background, 6 months graphic designer, 3 man-years programmers.
Number of Users	25
Status	Finished; internal use
Language	Smalltalk, MPW-C
Communications	TCP/IP using IPC package
Designers	Thomas Erickson, Gitta Salomon, Ruth Ritter
Organization	Apple Computer
Availability	Only internally to Apple ATG
Design goals	Serve as research platform for interface and architectural explorations. Allow ordinary users to create personalized information flows; support passive alerting, scanning and capture of information.
Used	Used in various internal tests; not available for the Internet experiment
Problems	Dealing with multiple servers and a directory of servers

Rosebud is a project within Apple Computer's Advanced Technology Group. Its principle objective is to serve as a platform for investigations into what is needed to make remote information accessible to ordinary Macintosh users. The investigations have two foci: human interface components and techniques; and system architecture issues. In this article we focus exclusively on the human interface aspects of Rosebud.

Rosebud is similar to the WAIS interfaces described here in that it uses the Z39.50 protocol to access multiple, remote database; it differs from them in that it contains extra underpinnings for making information access an integral part of the Macintosh environment. The Rosebud system does not currently provide access to the internet WAIS servers (for reasons of network security, rather than basic incompatibilities), and is not available for the internet experiment.

4.1 Design Rationale

The design of the Rosebud interface began with a study of the practices and problems of ordinary information users. The principle focus was on information users at KPMG Peat Marwick in San Jose, the original client site for WAIS; in addition, several groups of users of on-line information services within Apple were also studied (Erickson, 1991). Interviews with accountants at Peat Marwick enabled the designers to put together a schematic of how information (mostly paper-based information) flowed through their offices (figure 5).

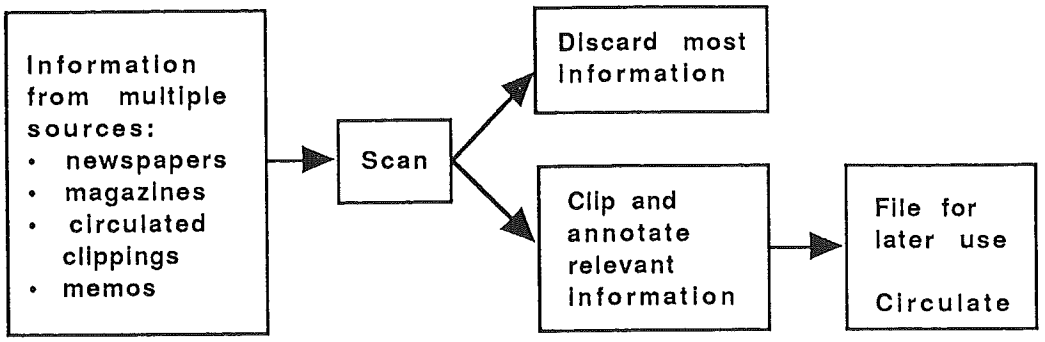


Figure 5 Information flow through accountants' offices.

Several features of this schematic informed the design of Rosebud. First, information typically came to the accountants via newspapers, magazines, and memos; instances where the accountants went out of their way to search for information were less frequent. Second, the accountants never talked about "reading" information; they always spoke of scanning, or skimming it—they didn't have time to read it. This suggested that a good interface should provide a way for the users to scan retrieved information quickly. Third, accountants remarked that they discarded most information, including information that might be useful. Potentially useful information was discarded for two reasons: the accountants didn't have the physical space to store everything, and they knew from experience that if they tried to save too much, they wouldn't be able to find anything later, when they actually needed it. This suggested that giving users access to remote information was just half the problem; users also needed tools for archiving, organizing, and re-retrieving information. Finally, when users did come across information that seemed worth saving, they would typically cut it out (the accountants used, almost exclusively, paper-based information), and then they would annotate it by circling, underlining, or jotting a few notes in the margin. Annotation turned out to be an important concept: not only did it help the user who annotated when the information was re-retrieved later on, but it also helped others scan the information more quickly when copies were passed on to them.

The consequence of these observations was a design for a system which allowed users to define topics of interest which would be automatically retrieved, and would then permit them to scan those items and save them into an environment where they could be annotated, organized and re-retrieved.

4.2 Human Interface

The Rosebud interface design has three components: reporters, newspapers, and notebooks. Reporters are for retrieving information. Users give reporters assignments which specify what to look for, and where to look. This is shown in figure 6: users enter words describing the information in which they're interested, check off the information sources they wish the reporter to search, and, if they so choose, automate the reporter so that it searches the databases on a daily or weekly basis. Upon pressing the "Search" button in the assignment window, a reporter is created, performs the search, and returns with a list of results (figure 7).

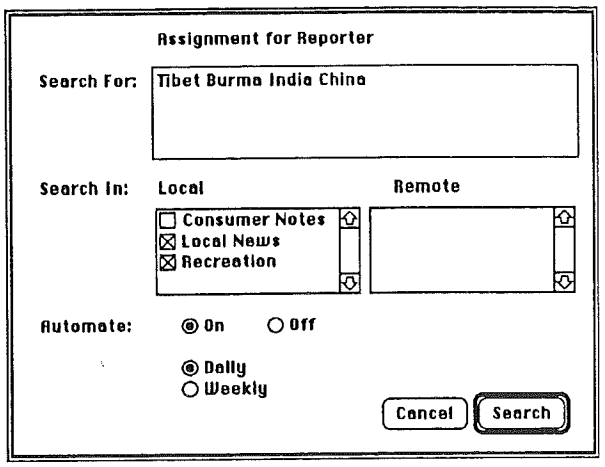


Figure 6 *Creating a reporter—the assignment window.*

The reporter window (figure 7) provides users with a variety of ways to look over their results, and refine their queries. The results are shown in the "Best Guesses" pane. (The name '"Best Guesses" was chosen to provide some indication that inaccuracy could be expected; our observations of users had shown that they were often mystified by some of the items that showed up as the results of searches.) The asterisks to the left of items indicate their relative relevance, and the pop up menu above the pane allows users to order the list by date or relevance. Simply selecting an item shows a preview of it—a short excerpt with search terms highlighted in boldface (figure 8). Previews are useful because users can get a look at a little bit of the item without incurring the overhead of downloading the whole article over the network. Users also have the options of saving articles to their disks or opening them for viewing. Finally, having looked over their results, users can refine their search in the bottom pane of the window.

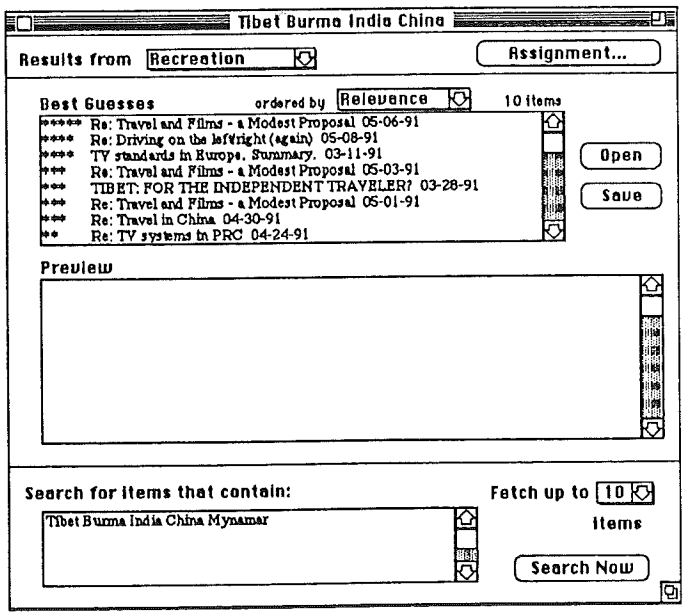


Figure 7 *The reporter window contains the results of the search and provides means for previewing, opening, and saving results.*

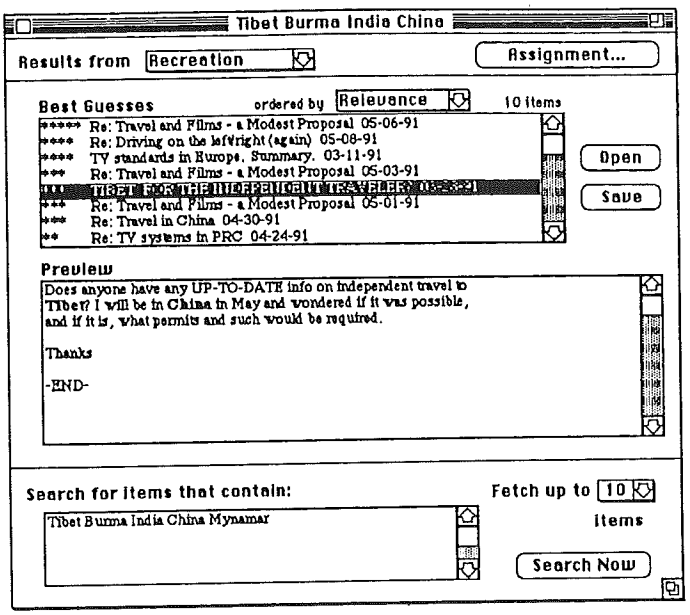


Figure 8 *The reporter window makes it easy to scan through hits. Clicking on a retrieved items generates a preview which shows an excerpt to the hit with the search terms (Tibet and China) highlighted in boldface. The user can refine the query in the lower pane of the window.*

The above sequence occurs whenever a user creates a new reporter. However, since users are likely to use many reporters, and because the initial user studies indicated that ways of skimming through incoming information were important to the accountants, the newspaper was provided to support rapid scanning of new information. The model of a newspaper is quite simple (figure 9): on the left is an index column which contains the names of all reporters, and to the right are two columns of news. Each reporter 'owns' one news column and publishes the title, date and an excerpt of each item in its column. The columns scroll independently, using 'minimalist' scroll bars to prevent the multiple scroll bars from visually overloading the screen. If an excerpt seems interesting, double clicking on it opens the full article in a window, from which it can be viewed, printed, or saved. Thus, rather than having to open up a dozen reporters every morning to see what's new, the user can go to one place, the newspaper.

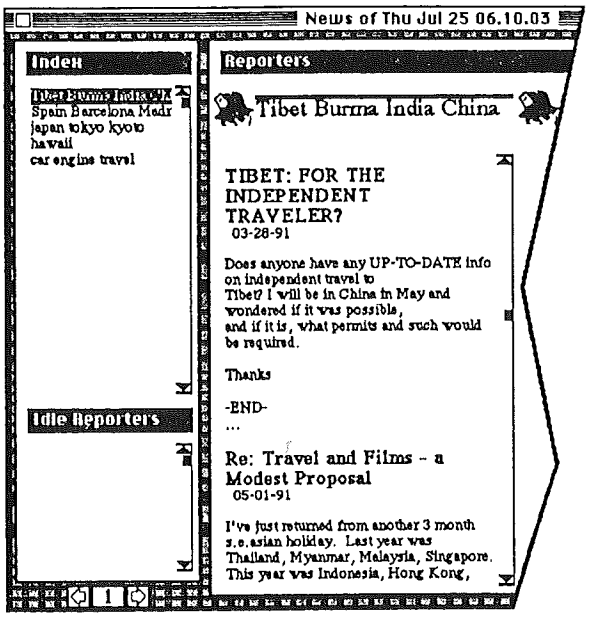


Figure 9 *The newspaper allows users to quickly scan through new items retrieved by the reporters which are working automatically.*

The newspaper can also serve as a control center for the Rosebud interface. The user can open a reporter by clicking on its name or icon at the top of its news column. Consequently, if a reporter's column has strayed from the desired topic, the user can quickly get to the reporter and revise its assignment. The index also lists inactive reporters (those either not automated, or that haven't found anything new since the last newspaper), so they too can be opened, and automated or otherwise adjusted.

A third component of the Rosebud interface—the notebook—was designed but not implemented. Notebooks are environments within which users may save, annotate, and organize retrieved information. Notebooks were designed in response to the observations of Peat Marwick accountants, which indicated the need for an environment which supported the way accountants worked—in particular, notebooks were intended to support annotation, and re-finding retrieved information at a later date. A particularly nice feature of the notebook design was its use of annotations as landmarks for re-finding information. The notebook design, and its rationale, is described in (Erickson, 1991).

4.3 Implementation

Rosebud consists of two parts: a user interface application written in SmallTalk (to facilitate the rapid changes in the interface necessary to effectively conduct interface design research), and a search manager application written in MPW C, which uses TCP/IP and an IPC package to communicate with search engines and remote databases and with the user interface application; like the other WAIS interfaces, Rosebud uses the WAIS protocol package to communicate. The human interface was designed for Macintosh II class machines, with 13 inch color screens. The search manager application runs in the background under MultiFinder, and is able to access information and construct newspapers while the user interface application is not running.

4.4 Observations and Testing Results

The Rosebud human interface was subjected to informal testing on 14 users. Users were told only that Rosebud was an application for finding information, and then given a particular topic to find information on. They were given no help or documentation. Note that although informal, this type of testing is very stringent, in that users approach the application knowing almost nothing about what it is, or why they would actually use it. Data collection consisted simply of recording their questions, observations, and problems as they went along, administering a post-test questionnaire, and then asking them a few, open-ended questions. Here are a few of the more general observations.

- Over 80% of those who tried the Rosebud interface responded very positively to it, and said that they would use something with its capacities as part of their daily work routine. Two thirds of users indicated that they would usually use newspapers to browse through information (instead of reporters).
- At the end of the test, over two thirds of the users said they liked the metaphors of reporters and newspapers; however, almost all users had some difficulty in getting started. The typical problem was that users did not associate reporters with a way of retrieving information. When asked to find information, users first looked for an item called search; when they didn't find this, they usually turned to the newspaper, which is, in fact, where they look for information on a daily basis. It is possible that this problem can be remedied by minor interface changes (e.g. putting a "New Reporter" item in a search menu); alternatively, it may be that the metaphor is inappropriate.
- A number of users were lead astray because they had conceptual models of information retrieval based on their familiarity with query languages and structured databases. Such users tended to be wary of entering search terms because they weren't sure of what the appropriate syntax was, and didn't understand what "relevance" meant. Those that did know what relevance was wanted to know how the information server calculated it.
- Users liked previews a lot—especially the feature of highlighting keywords in boldface. They wanted to see boldface keywords in the newspaper and article windows. Users also wanted the ability to select text in the newspaper and article windows and change the style or font themselves, so that they could annotate significant items. This parallels practices observed in our initial observations of accountants, where we found that annotation plays several important roles.
- A variety of low level interface problems, due to terminology or graphic design were discovered. Some examples: users did not usually recognize the asterisks in the "Best Guesses" window as indicators of relevance; users didn't think that "idle reporters" was a good name, and said that it was very important to distinguish between reporters which had found nothing, and those which weren't looking.

As of this writing, the next phase of Rosebud human interface testing is about to begin. In this phase, a small set of users will be observed over the course of a month, in which they have the option of using Rosebud from their desktop machines to access meaningful data. This phase of testing will allow a more realistic assessment of Rosebud, in that it will last long enough to permit users to build up their own set of reporters, and to access newspapers which contain information of personal import.

5. X WINDOWS BASED INTERFACE FOR WAIS: XWAIS

XWAIS At A Glance	
Target Machine	X-windows terminals on unix machines
Effort	4 man-months
Number of Users	500
Status	finished, freely distributed
Language	C
Communications	TCP/IP
Designer	Jonathan Goldman
Organization	Thinking Machines
Availability	Available anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Copy WAIStation so that we can leverage one design, portable and based only on freeware Display data in many different formats (image, text, etc)
Used	Used in the internet experiment Heavy use by X users within Thinking Machines and outside
Problems	Installing it has caused many users to stumble. The number of variables (architectures, X directory structures) makes it difficult to make it portable touch on the ability to handle different types (this is unique to this interface). uses other programs to help (like interapplication communication)

The WAIS interface for the X Windows environment was developed for the Internet experiment to provide an X Windows based interface for a growing community. It was built to look as much like the Macintosh WAIS interface (WAIStation) as possible, given the limitations of the freely distributed X Windows software. Since the metaphors in XWAIS are nearly the same as those for WAIStation, a user of one system can easily move to the other, without having to learn much new. In fact, the underlying data structures are identical to those in WAIStation, so questions can be copied from a Macintosh to a UNIX machine running XWAIS, and used without modification.

XWAIS supports interactive WAIS access, including question entering, source selection, addition of relevant documents and pieces of documents. Unlike WAIStation, XWAIS retrieves an entire document when requested, instead of just the parts being viewed. We decided this was acceptable, since the underlying networks for X will most likely be fast.

Since XWAIS runs under X windows, and was built for the UNIX operating system, it can take advantage of the tools available for these systems to display a wide range of document formats. A simple filter interface is provided in the application (as an X resource) to allow a user to select the tool required for a given type of document, e.g, if the document is a postscript file, xps can be used to view it. This is a feature that is not available in any of the other user interfaces described here.

In order to distribute this software without restriction, XWAIS uses the freely distributed Athena Widget set included in the X11R4 release from MIT. Although these widgets don't look as nice as some others that are available, they can be used to build a useful interface. Some aspects of this interface are restricted by the nature of the widgets available. XWAIS was built using the Xt X Toolkit Intrinsics, and allows a large amount of customization of the appearance of the display using X resources. The application relies heavily on the Xt resource mechanism, and will not run unless these resources are in place. The "object-oriented" feel of these widgets made building the interface rather easy, once the widget with the closest desired functionality was found. Finding the correct widget was the hardest part. Most of the actual behavior of the interface is controlled by "call-backs" - the methods that widgets inherit.

The XWAIS application is actually two separate applications: XWAIS, a simple shell for selecting sources and questions, and xwaisq, the application that actually performs WAIS transactions. The C code in xwaisq is also used

in waisq, the shell-support program for GNU Emacs WAIS. This allows users to use simple UNIX facilities to submit questions created by xwaisq using waisq (e.g. a crontab entry to periodically query a server).

The implementation for XWAIS was done in C (6k lines), using the X11R4 release of X windows from MIT, the Xt X Toolkit Intrinsics, and the Athena Widget Set, included in the X Windows release.

XWAIS is a text-based user interface built in a graphical window environment. Some additional graphical metaphors would be desirable, but the limited widget sets precluded that. It would take a considerably larger amount of work to add much graphics to this application. Perhaps some other X toolkit would provide simpler methods for doing this.

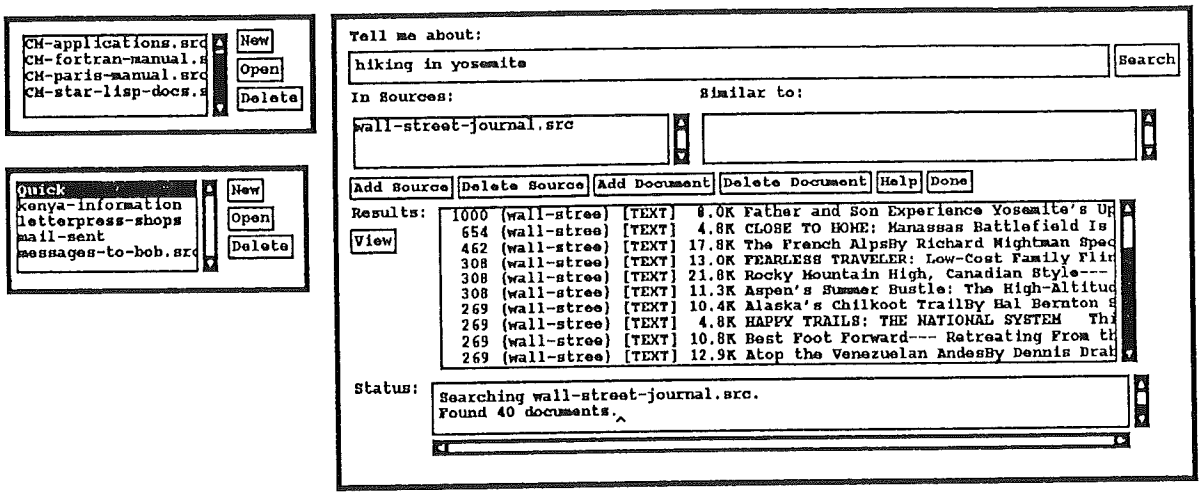


Figure 10 The XWAIS interface, including the Questions and Sources windows, and an open question.

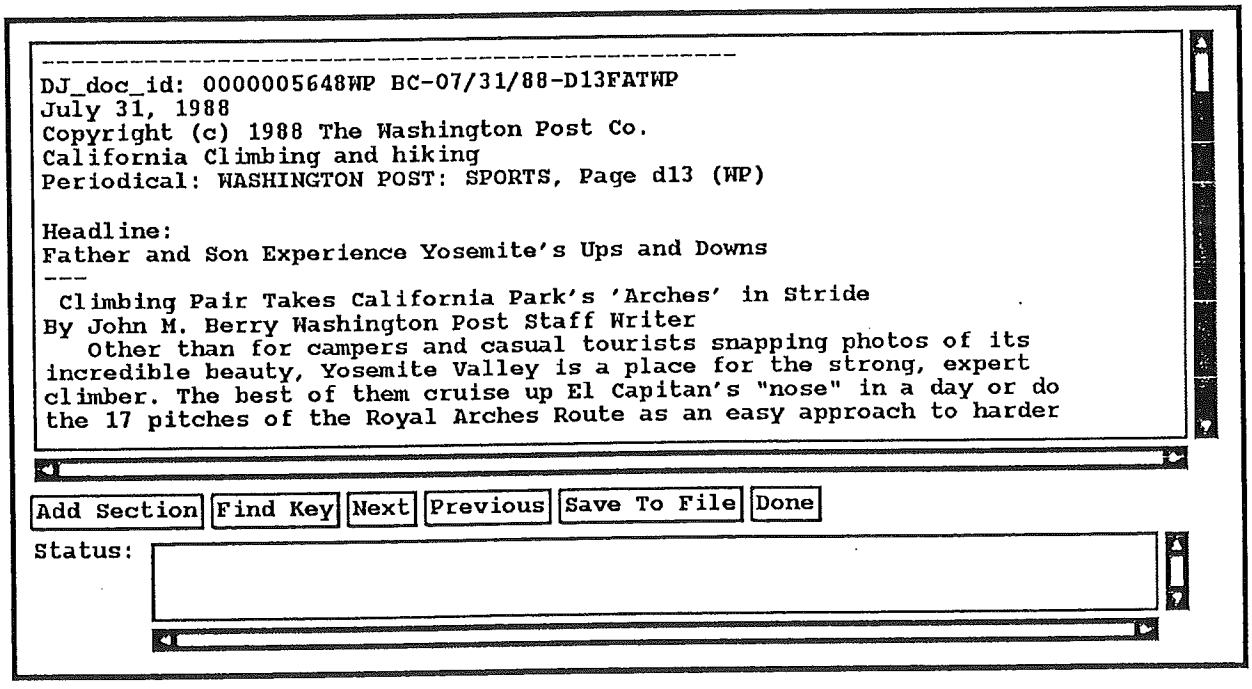


Figure 11 A document displayed in the XWAIS interface.

6. GNU EMACS WAIS INTERFACE: GWAIS

GWAIS At A Glance	
Target Machine	terminals on unix machines
Effort	2 man-months
Number of Users	500
Status	finished, freely distributed
Language	gnu-lisp, and C
Communications	TCP/IP
Designer	Jonathan Goldman
Organization	Thinking Machines
Availability	Available anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Copy WAIStation so that we can leverage one design, Use precedent from other gnu-emacs applications: RMAIL, dired
Used	Used in the Internet experiment with heavy use by some gnu-emacs users
Problems	Dealing with the directory of servers. Using passive alerting

The WAIS interface on GNU-Emacs/Unix (GNU) was developed specifically for the Internet experiment for a technically strong user population. The reasons it was developed were: the large number of emacs users, the extensibility, the ubiquitous nature of character display terminals, and the component nature of emacs which meant WAIS could be integrated into email, bboards, and programming tools.

The design of the interface was a cross between WAIStation and other emacs interfaces. The direct manipulation of WAIStation was replaced by command keys, as is common in emacs applications. The choice of command keys were modeled on the dired and RMAIL emacs applications.

GWAIS allows users to access the interactive features of WAIS: question entering, relevance feedback, displaying document, and source selection. An extra feature, not found in the other interfaces, is an interface to an indexer for creating sources, but it appears that this feature is not heavily used. Furthermore it allows questions to be saved, but it depends on the user to automate the update of questions and sources using cron or other Unix tools. Graphic documents can be displayed on X Windows terminals if the user has set up the environment variables.

The implementation of GWAIS was in emacs lisp (2K lines) and in C code (3K lines). About half of the time of a typical search and retrieval is spent in reading the data into lisp.

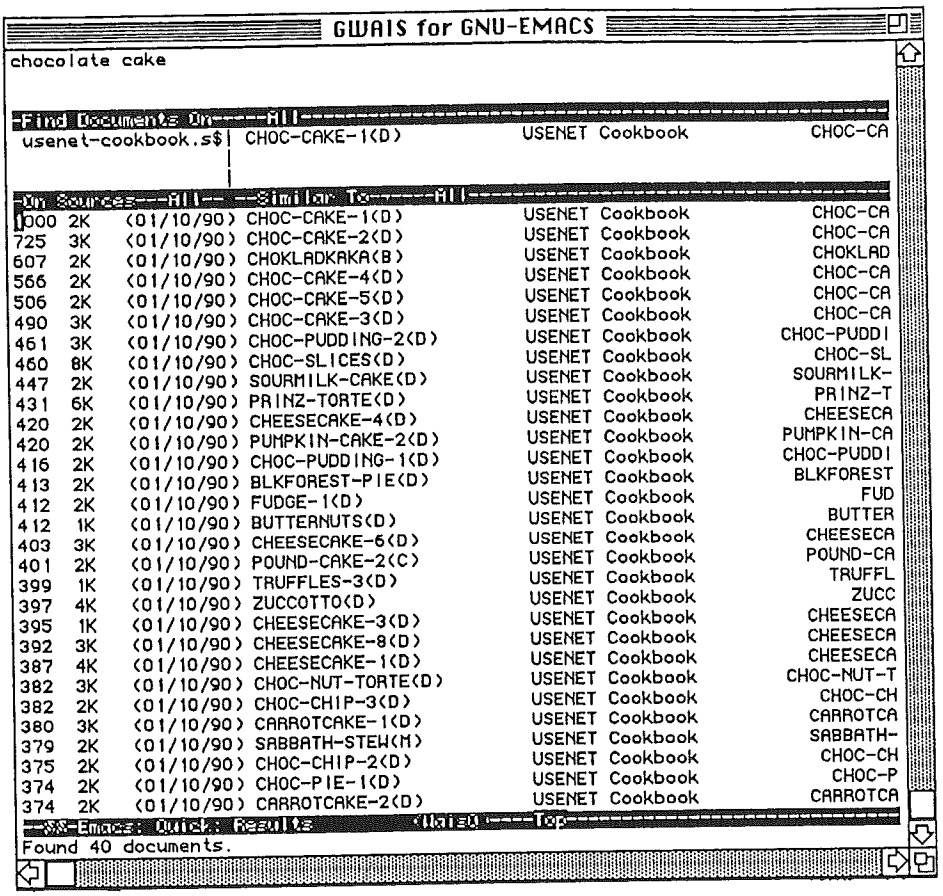


Figure 12 The GWAIS interface, displaying the results of a relevance feedback search.

7. SCREEN BASED (TERMINAL) WAIS INTERFACE: SWAIS

SWAIS At A Glance	
Target Machine	Terminals connected to Unix systems
Effort	1 man-month
Number of Users	900
Status	beta
Language	C
Communications	TCP/IP
Designer	John Curran
Organization	NSF Network Service Center
Availability	To be included in WAIS release, anonymous FTP from /public/wais/wais*.tar.Z@think.com
Design goals	Highly Portable, Provide straight-forward user interface, Utilize existing application key mappings (rn, vi, emacs), Support multiple servers per query, Allow for personal "source" directory and a common source directory, Allow for useful source discovery via searches, Provide simple active tool with little state (no question storage, relevance feedback, or passive notification)
Used	Internet users via telnet: k-12 students, educators, user services staff, librarians, and (occasionally) network staff
Problems	Dealing with the directory of servers. Lack of information in many server-returned records. Providing simple and uniform nomenclature Planning for large numbers of sources.

To open WAIS to a wider community of users, an interface was developed to run on dumb terminals or over telnet sessions. It is called "SWAIS" for Screen WAIS since it is uses a character display terminal screen for the interface. The user communities that this interface can serve are dial-in users, telnet users, and low-end terminal users.

The design of the interface involved 3 screens: a single screen listing all known servers that the user could pick from; a list of search result documents headlines; and a document display screen. Listing all servers and allowing users to pick which servers to use encourages users to ask questions of multiple servers. Unlike the other interfaces, the sources list shows what site runs it and how much it costs (if anything). The resulting document screen includes headlines and how many lines it is, but its innovation is to show what source it came from.

It does not handle relevance feedback or downloading new sources from the directory of servers. Another drawback is using it with large numbers of sources since moving around the list requires scrolling. On the other hand, this server has proven to be very popular on the Internet because of its ease of use, all a user has to do is telnet to a specific machine to use it.

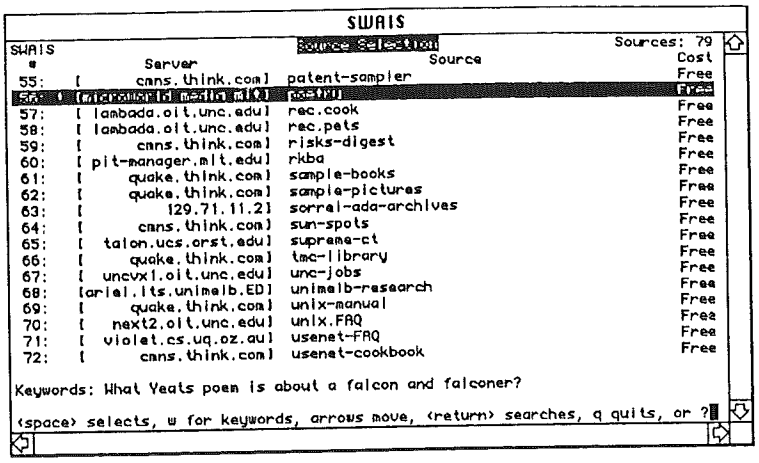


Figure 13 *The SWAIS query building screen. The poetry source is selected, and search terms are entered. This interface does not currently support relevance feedback.*

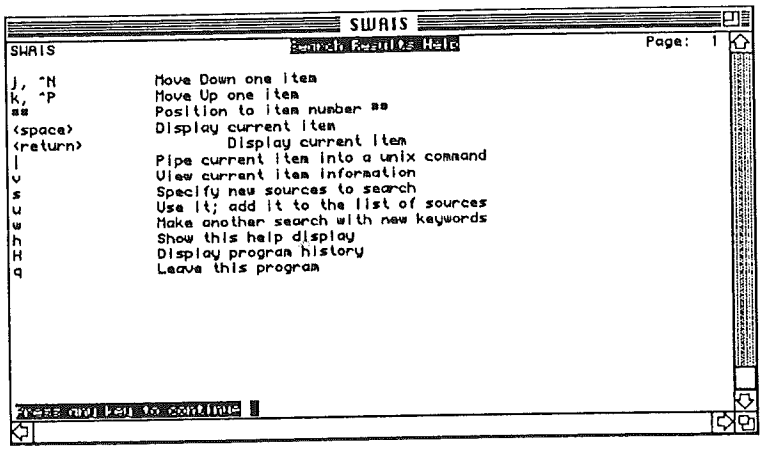


Figure 14 *The SWAIS help screen.*

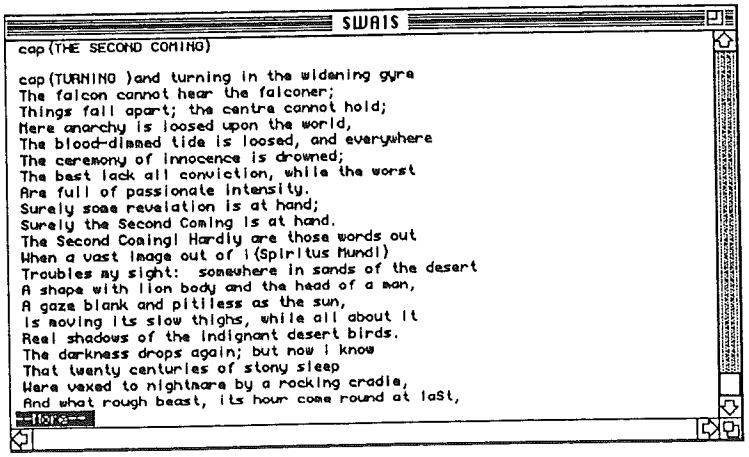


Figure 15 *A document displayed in SWAIS.*

8. CONCLUSION

This paper has described five interfaces developed in the context of the Wide Area Information Servers project. The interfaces presented here were developed with different constraints in mind, so it is not useful to compare them directly; instead they may serve as examples of differing responses to issues such as screen size, workstation power and intelligence, communication speeds, and user needs and practices.

The interfaces designed so far have addressed some of the critical issues for end-users to accomplish interactive searches in a wide area network. These include ways of finding which information servers contain relevant information, supporting searching by ordinary users, and supporting browsing of, and passive alerting about, newly retrieved information. The alerting aspects of the interfaces have not been tested much in this environment due to the lack of appropriate data sources for this type of searching. It is probably fair to say that any of the design solutions described here can be improved upon by further work.

The WAIS Internet experiment has revealed a number of issues requiring further work. In the Internet environment we have observed (in the logs of user queries) that users have a difficult time finding out what is in a database, thus demonstrating that there is a lack of browsing or scanning facilities in the interfaces, protocol, and servers, as well as a general shortage of descriptive information about databases.

Finally, there are a variety of other issues raised during the studies of the Peat Marwick accountants which have received little or no work. Document layout is one such problem. Accountants mentioned that sometimes they want to retrieve documents not because of the information they contain, but to look at their layouts (accountants will often examine successful proposals to a client when preparing a new proposal). More generally, users regard pictures, diagrams, tables, and charts as essential components of a document's content. Unfortunately, support for different document formats, and for the retrieval and display of non-textual information within them is very limited on most existing clients.

Another issue is called the boilerplate problem. Accounting documents often contain a large amount of boilerplate, standard text which varies little from document to document. What tools are needed to allow users to effectively retrieve, order, and browse a large set of documents which are 95% similar? Note that boilerplate is characteristic of a wide variety of business proposals and legal documents, not just accounting documents. In fact, the analog to boilerplate occurs in scientific documents in which standard terms and descriptions are used to describe procedures and methods used in an investigation.

A number of other issues remain to be addressed. Users are very interested in being able to see what queries other users are conducting, and what information servers and articles are most popular. A frequent suggestion is to allow users to rate the 'goodness' of articles they retrieve. However, in a commercial setting, information about the kind of questions being posed by a particular company or person can be revealing and valuable. Clearly, the utility that such information could provide must be balanced by concerns about confidentiality and privacy, and mechanisms for user control of descriptive information are essential. Other issues include how to control the pricing, copyright, and distribution issues which accompany 'for-pay' information.

In summary, there is an immense amount of work to be done. A central part of this work involves further research and development of interfaces. We hope that designers will find that WAIS—with its common protocol and defined infrastructure—can serve as a platform from which to pursue these, and other, research issues.

FOR MORE INFORMATION ON WAIS

The success of a WAIS-like system depends on a critical mass of users and information services. In order to encourage development and use, Thinking Machines is making the source code for a WAIS protocol implementation freely available. While this software is available at no cost, it comes with no support. We hope that it will facilitate others in developing servers and clients.

For more information, please contact:

Barbara Lincoln (barbara@think.com)

Thinking Machines Corporation
1010 El Camino Real, Suite 310

245 First Street

Menlo Park, CA 94025
415-329-9300

Cambridge, MA 02142
617-234-1000

REFERENCES

(Davis, 1990) F. Davis, B. Kahle, H. Morris, J. Salem, T. Shen, R. Wang, J. Sui, and M. Grinbaum, "WAIS Interface Protocol Functional Specification," Thinking Machines Corporation, April 1990. Available via anonymous ftp: /pub/wais/doc/wais-concepts.txt@quake.think.com or wais server wais-docs.src

(Dongarra, 1987) J. Dongarra and E. Grosse, "Distribution of Mathematical Software Via Electronic Mail," CACM, 1987, Vol. 30, pp. 403-407.

(Egan, 1989) D. Egan, J. Remde, L. Gomez, T. Landauer, J. Eberhardt, and C. Lochbaum, "Formative Design-Evaluation of SuperBook," ACM Transactions on Office Information Systems, January 1989.

(Erickson, 1991) T. Erickson and G. Salomon, "Designing a Desktop Information System: Observations and Issues," Proceedings of the ACM Human Computer Interaction Conference, 1991. ACM Press, April 1991, pp. 49-54.

(Ginther-Webster, 1990) K. Ginther-Webster, "Project Mercury," AI Magazine, 1990, pp. 25-26.

(GNU) For information on the Free Software Foundation and the GNU project, see: /pub/gnu/*@prep.ai.mit.edu. For emacs see: /pub/gnu/emacs-*.tar.Z@prep.ai.mit.edu.

(Kahle, 1991a) B. Kahle and A. Medlar, "An Information System for Corporate Users: Wide Area Information Servers," April 1991. Online Magazine, September 1991.

(Kahle, 1991b) B. Kahle, J. Goldman, H. Morris, T. Shen, "Electronic Publishing Experiment on the Internet - Wide Area Information Servers" In preparation.

(Malone, 1986) T. Malone, K. Grant, and F. Turback, "The Information Lens: An Intelligent System for Information Sharing in Organizations; In Human Factors In Computing Systems," CHI'86 Conference Proceedings, Boston, MA; ACM, New York, 1986, pp. 1-8.

(Mosis) For more information on the Mosis fabrication facility write to mosis@mosis.edu.

(NeXT) NeXT Computer Inc., 900 Chesapeake, Redwood City, California, 94063, (415) 366-0900.

(NISO, 88) "Z39.50-1988: Information Retrieval Service Definition and Protocol Specification for Library Applications," National Information Standards Organization (Z39), P.O. Box 1056, Bethesda, MD 20817. (301) 975-2814. Available from Document Center, Belmont, CA. Telephone (415) 591-7600.

(ON Technology) ON Technology Inc., 155 Second Street, Cambridge, Massachusetts, 02141, (617) 876-0900.

(VERITY) Verity Inc., 1550 Plymouth Street, Mountain View, California, 94043, (415) 960-7600.

ACKNOWLEDGMENTS

The Rosebud interface was designed by Gitta Salomon, Tom Erickson, and Ruth Ritter. Kevin Tiene had significant impact on the interface design, and also implemented the whole thing. Kevin Tiene, Charlie Bedard, David Casseres, and Eric Roth designed and implemented the search manager, and other underpinnings of Rosebud. Steve Cisler and Janet Vratny-Watts, of the Apple Library, provided valuable information on the state of the on-line information universe, and on the needs of information end users.

At Thinking Machines: Ottavia Bassetti, Franklin Davis, Patrick Bray, Danny Hillis, Rob Jones, Barbara Lincoln, Gordon Linoff, Chris Madsen, Gary Rancourt, Sandy Raymond, Tracy Shen, Craig Stanfill, Steve Schwartz, Robert Thau, Ephraim Vishniac, David Waltz, and Uri Wilensky.

At the National Science Foundation: Suzi, Alex, et. al.

Session II: Z39.50 Protocol and Wide Area Information Servers

C. Lynch

The Z39.50 Protocol: Questions and Answers

Clifford A. Lynch
Division of Library Automation
University of California Office of the President
300 Lakeside Drive, 8th floor
Oakland, California 94610-3550
415/987-0522
CALUR@UCCMVSA.BITNET or CALUR@UCCMVSA.UCOP.EDU

1. What is Z39.50?

Z39.50 is an American National Standard that was approved in 1988 by the National Information Standards Organization (NISO), an American National Standards Institute- (ANSI) accredited standards writing body that serves the library, information, and publishing communities. Work is currently well along on a revised version that (hopefully) will be adapted in 1992.

Z39.50 is an applications-layer protocol within the OSI reference model developed by the International Standards Organization (ISO). Its purpose is to allow one computer operating in a client mode to perform information retrieval queries against another computer acting as an information server.

The standard provides a uniform procedure for client computers to query information resources such as server computers supporting online library catalogs. For example, the development of a client program running on one machine may provide end users with a common means of access to a variety of information resources attached to a computer network.

2. Does Z39.50 apply only to bibliographic data, or can other types of information be accommodated?

While many of the initial applications of Z39.50 are for use with bibliographic data (online public access library catalogs, for example), the protocol is actually quite general, and search attribute sets can be defined which allow the protocol to work with most other types of data. Some specialized data, such as images, may necessitate protocol extensions. Various groups are currently exploring the area of protocol extensions, and are defining attribute sets for other types of data, such as full text.

3. Is Z39.50 an International Standard?

No. Z39.50 is an American National Standard. However, there is an ISO standard called Search and Retrieval, ISO 10162/10163 (service and protocol documents, respectively), which was formally adopted to International Standard (IS) status in 1991. SR is almost identical to a subset of Z39.50. Z39.50 is being revised to harmonize with SR, at which point the American National Standard will be a compatible superset of the International standard.

4. I have a GEAC (or any other) circulation system and a Carlyle (or any other) online public access catalog at my library. Will Z39.50 enable them to talk with each other?

In theory, Z39.50 provides a basis for allowing such communication. However, attribute sets and data transfer formats appropriate to queries of circulation systems by public access catalogs have not yet been defined, much less tested in practice.

The National Information Standards Organization's Technical Plan, which defines NISO's standards development agenda, calls for work in this area.

In addition, for such a link to work, both vendors not only would have to implement Z39.50 but they would have to integrate the protocol implementation with their overall software applications to create an effective intersystem link.

In a joint project with Data Research Associates (DRA), Digital Equipment Corporation (DEC), and the University of California, a link will be developed between the DRA system being installed at UC Davis and the MELVYL system, which would allow the MELVYL system to perform Z39.50-based searching of the circulation status file on the DRA system. This searching would display information on the availability of materials as part of a MELVYL catalog display for material held at the Davis campus.

5. Can I buy a Z39.50 package?

Today, I know of no organization that is marketing a Z39.50 client or server, although a number of the organizations that are doing work in this area may be willing to share code. In the longer term, Z39.50 "products" will fall into two categories. There will be client implementations that run on workstations or mainframes, and it is likely that at some point one may see commercial products of this type that provide a user interface to many Z39.50 servers. It seems much less likely that we will see Z39.50 server products that stand alone since a Z39.50 server becomes an integral part of an information retrieval application. We may see vendors of information retrieval systems or database management software offering Z39.50 toolkits or support as part of their overall offerings. And it is likely that we will see vendors of library-specific applications, such as the online library catalogs offered by DRA and NOTIS, integrating Z39.50 server and/or client support in their products.

6. What vendors are currently working on Z39.50? What other efforts are underway?

There are a number of efforts underway; more are starting up all the time. Here are some of the more elaborate projects that we know about as of early 1992:

•OCLC is working on Z39.50 server and client support for a range of systems, including EPIC. OCLC has also done work with NYSERNet in Z39.50. Contact: Ralph LeVan (OCLC, 6565 Frantz Rd., Dublin, OH 43017; (614)761-6115; rrl@rsch.oclc.org).

•RLIN is working on Z39.50 server and client support, and Z39.50 is very much part of their strategic technical plan. Contact: Madeline (Lennie) Stovel (RLG, Inc., 1200 Villa St., Mountain View, CA 94041-1100; (415)691-2259; bl.mds@rlg.stanford.edu).

•Data Research Associates (DRA), a library automation vendor, is developing Z39.50 support (client and server) for their product. They are participating in a joint project with Digital Equipment Corporation, and the University of California President to build a link between a DRA system at UC Davis campus and MELVYL. This link would allow UC's MELVYL system

to perform Z39.50-based searching of the circulation status file on the DRA system. Contacts: Clifford Lynch (University of California Office of the President, 300 Lakeside Dr., 8th floor, Oakland, CA 94610-3550; (415)987-0522; calur@uccmvsa.bitnet). Mary Heath (University of California at Davis, Shields Library, Davis, CA 95616; (916)752-0129). Jim Michael (DRA, 1276 N. Warson Rd., St. Louis, MO 63132-1806; (800)325-0888; jim@dranet.dra.com).

•The University of California Division of Library Automation is developing client and server Z39.50 support with help from Digital Equipment Corporation in conjunction with Pennsylvania State University. This is both for the UC IBM 3090-based MELVYL online catalog and the VAX 9000-based Pennsylvania State University LIAS, as well as for selected microcomputers. Contacts: Clifford Lynch (see above). Eric Ferrin (Pennsylvania State University, E8 Pattee Library, University Park, PA 16802; (814)865-1818; egf@psulias.bitnet).

•Project Mercury is a joint Carnegie-Mellon University/OCLC/Digital Equipment Corporation project to develop an electronic library at CMU. It is using Z39.50 as a central element in linking clients and servers. Contact: Bill Arms (Carnegie-Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213; (412)268-2122; wya@andrew.cmu.edu).

•UC Berkeley is developing an Information Server Project to coordinate campus community information. The Berkeley Information Server will use Z39.50 in a client-server architecture =, with UNIX servers and initial clients targeted for workstations and personal computers. Contact: Margaret Baker (University of California, 289 Evans Hall, Berkeley, CA 94720; (415)642-5601; margaret@garnet.berkeley.edu).

•The Linked Systems Project is organized by the Library of Congress, OCLC, and RLIN, which are currently interchanging bibliographic and authority records using a predecessor protocol to Z39.50 over an OSI network, with plans to upgrade to current Z39.50 standards. Contact: Sally McCallum (Library of Congress, Room 327, 10 First Street, SE, Washington, DC 20540, (202)707-5807; bb.shm@rlg.bitnet).

•The Florida Center for Library Automation is working on the development of Z39.50 server and client functions for the NOTIS library automation system. They have received a US Department of Education Title II-D grant for the project. Contact: Mark Hinnebusch (Florida Center for Library Automation, Suite 320, 2002 NW 13th St., Gainesville, FL 32609; (904)392-9020; fclmth@nervm.bitnet).

•Thinking Machines, Dow Jones, and Apple Computer are working on the development of a prototype information server for a Dow Jones database based on a somewhat extended and modified Z39.50 protocol. Contact: Brewster Kahle (Thinking Machines, 898 14th St., San Francisco, CA 94114; (415)863-6485; brewster@think.com).

•There is an active implementors' group for Z39.50 which meets every few months, chaired by Mark Hinnebusch (Florida Center for Library Automation, Suite 320, 2002 NW 13th Street, Gainesville, FL 32609; (904)392-9020; fcla@nervm.bitnet). They run a network mailing list called Z3950IW at nervm.nerdc.ufl.edu and nervm.bitnet. Contact: Mark Hinnebusch for subscription information. An FTP file server containing Z39.50 documents is also available.

•A Z39.50 testbed is being established under the auspices of the Coalition for Networked Information (CNI). This group, which includes about 15 organizations, plans to develop and

demonstrate multiple interoperable Z39.50 implementations running over the Internet within a short time frame. Contact: Clifford A. Lynch (see above).

7. When can I see a Z39.50 implementation?

There are several running prototypes today, such as CMU Project Mercury, which was shown at EDUCOM '89 in Ann Arbor. It seems likely that you will see demonstrations at meetings of groups such as the American Library Association, NET '92, and EDUCOM during 1992.

8. How do I get a vendor to write a Z39.50 interface?

Today, most library automation local system vendors do not seem to have firm plans to implement Z39.50. You need to make it clear to vendors and potential vendors that this is an important feature for the future, and to consider needs for Z39.50 in procuring new systems.

9. How do I get a copy of the Z39.50 standard?

Order from Transaction Publishers, Rutgers — The State University, New Brunswick, NJ 08903 (201) 932-2280. Transaction publishes NISO standards. You can contact NISO at National Bureau of Standards, Administration 101, Library E-106, Gaithersburg, MD 20899, NISO@nbsenh.bitnet.

You may also want to track ISO 10162/10163 (available from Omnicom Information Service, 115 Park Street, SE, Vienna, Virginia 22180 (703) 281-1135), and subscribe to the Z39.50 implementors' mailing list (see above). The Library of Congress is serving as the maintenance agency for Z39.50 on behalf of NISO. Contact: Sally McCallum, Network Development Office/Processing Services, Library of Congress, Room 327, 10 First Street, SE, Washington, DC 20540, (202) 707-6237.

The following books and papers may also be of interest for more information. The paper by Lynch and Preston contains a much more extensive bibliography.

Henriette D. Avram. "The Linked Systems Project: Its Implications for Resource Sharing," *Library Resources and Technical Services* (January/March 1986), 30:1, pp. 34–46.

Henriette D. Avram. "Toward a Nationwide Library Network," *Journal of Library Administration* (Fall/Winter 1987), 8:3/4, pp. 96–115.

Henriette D. Avram. "LSP and Library Network Services in the Future," *EDUCOM Bulletin* (Summer/Fall 1988), 23:2/3, pp. 52–58.

Henriette D. Avram. "Building a Unified Information Network," *EDUCOM Bulletin* (Winter 1988), 23:4, pp. 11–14.

Michael Buckland and Clifford Lynch. "National and International Implications of the Linked Systems Protocol for Online Bibliographical Systems," *Cataloging and Classification Quarterly* (Spring 1988), 8:3/4, pp. 15–33.

Wayne E. Davison. "The WLN/RLG/LC Linked Systems Project," *Information Technology and Libraries* (March 1983), 2:1, pp. 34 – 46.

Ray Denenberg. "Linked Systems Project, Part 2: Standard Network Connection," *Library Hi Tech* (1985), 3:1, Issue 10, pp. 71 – 79.

Ray Denenberg; Bob Rader; Thomas P. Brown; Wayne Davison; and Fred Lauber. "Implementation of the Linked Systems Project: A Technical Report. Part One: Library of Congress. Part Two: The Western Library Network. Part Three: The Research Libraries Group. Part Four: OCLC," *Library Hi Tech* (1985), 3:3, Issue 11, pp. 87 – 107.

Judith Fenly and Beacher Wiggens (eds.). *The Linked Systems Project: A Networking Tool for Libraries* (Dublin, OH: OCLC, Inc., 1988).

Clifford A. Lynch. "Library Automation and the National Research Network" *EDUCOM Review* (Fall 1989), 24:3, pp. 21 – 26.

Clifford A. Lynch. "Access Technology for Network Information Resources," *CAUSE/EFFECT* (Summer 1990), pp. 15 – 20.

Clifford A. Lynch. "Information Retrieval as a Network Application," *Library Hi Tech* No. 4, Issue 32 (1990), pp. 59-74.

Clifford A. Lynch. "The Z39.50 Information Retrieval Protocol: An Overview and a Status Report," *Computer Communication Review* 21:1 (January 1991), pp. 58 – 70.

Clifford A. Lynch. "The Client-Server Model in Information Retrieval," *Interfaces for Information Retrieval* (Martin Dillon, ed.) (Westport, CT: Greenwood Press, 1991); pp. 301 – 318.

Clifford A. Lynch and Cecilia M. Preston. "Internet Access to Information Resources," *Annual Review of Information Science and Technology (ARIST)* Volume 25 (New York, NY: Elsevier, 1990), pp. 263 – 312.

Sally H. McCallum. "Linked Systems Project, Part 1: Authorities Implementation," *Library Hi Tech.* 1985; 3(1), Issue 10: 61 – 68.

Michael J. McGill. "Z39.50 Benefits for Designers and Users," *EDUCOM Review* (Fall 1989), 24:3, pp. 27 – 30.

Michael J. McGill; Larry L. Learn; and K.G. Thomas. "A Technical Evaluation of the Linked Systems Project Protocols in the Name Authority Distribution Application," *Information Technology and Libraries* (December 1987), 6:4, pp. 253 – 265.

10. Why should I care about Z39.50? Why is it important to my library? If it's so important why haven't I heard more about it?

Z39.50 is important because it is the best technology we have today to peit a single user interface to access the multiplicity of information resources becoming available on the national network.

The need for Z39.50 is just now becoming clear to a larger community as network information is becoming a more serious issue. You might want to become involved in the overall movement towards network information resources. Write the ARL/EDUCOM/CAUSE Coalition for Networked Information (Paul Peters, Director, 1527 New Hampshire Ave., NW, Washington, DC 20035; (202) 232-2466; paul@cni.org).

The Z39.50 Information Retrieval Protocol: An Overview and Status Report

Clifford A. Lynch
Director, Division of Library Automation
Office of the President, University of California
Oakland, California 94612-3550

calur@uccmvsa.bitnet or calur@uccmvsa.ucop.edu

Introduction

Z39.50, the computer-to-computer information retrieval protocol [1], is an OSI application layer protocol, which became a U.S. national standard in 1988. Although the Z39.50 protocol is reasonably well-known within the library and information services community (which was the primary mover in its development) and is now becoming a cornerstone in library automation strategy at a number of major academic institutions, it does not seem to be widely known or well understood in the broader computer networking community. This is particularly unfortunate today, as networked information resources are taking on an ever-increasing importance as key computer network applications [2]. The growing importance of the Z39.50 protocol, and the recent publication of Schoffstall and Yeong's article [3] in this journal, which provided a critique of some of the design decisions in Z39.50 without providing much background on the development of the protocol, its operation, or its current status, and which contained a number of factual errors, convinced the author of this paper that the time for an overview of the Z39.50 protocol and its current status for the computer networking community was overdue.

The paper first discusses the development and current status of Z39.50 as a standard, as well as the context in which this standards development occurred. The next section covers the functions of the protocol.[1] Z39.50, while it represents an important starting point, does not provide all of the functions necessary to support the full range of possible information retrieval applications today. Thus, the section on protocol function also discusses some of the shortcomings of the current standard, and work now underway to address these shortcomings by extending the protocol. The final section concludes the paper by commenting on some misconceptions about the standard.

This paper is not intended to be definitive; in particular, it omits discussion of several technical issues that pertain mainly to the application of Z39.50 in certain library automation settings. More details can be found in the references.

History and Status of the Z39.50 Standard

Work on information retrieval protocols has been ongoing since the mid-1970s. This work was initially funded by the Council on Library Resources, the then-National Bureau of Standards, and the National Commission on Library and Information Science; the Library of Congress played the lead role in organizing the effort. The basic concern, initially, was linking a small number of enormous files of bibliographic records housed at organizations such as the Library of Congress,

[1] Most of the section on protocol function is adapted, with permission, from a recent paper [4].

the Online Computer Library Center (OCLC), and the Research Libraries Information Network (RLIN) to create what was in effect a logical national information resource of bibliographic holdings [5, 6].

By the early 1980s this work had coalesced into an effort called the Linked Systems Project, which involved OCLC, the Library of Congress, the Research Libraries Information Network, and the Western Library Network (WLN), to build a prototype OSI network for support of a production application (the authorities application described in [4]) using a group of library applications protocols (including a precursor to Z39.50) running over this network. Since the OSI protocols were far from complete at that time, a series of working agreements were developed around then-current drafts of the relevant OSI protocols in order to permit implementations to proceed. By about 1984 the emphasis of the Linked Systems Protocol had shifted to implementation, and the draft information retrieval protocol (see [7]) was passed to Committee D of the National Information Standards Organization (NISO), the ANSI-accredited standards development body for libraries, information services, and publishing, for further refinement and consideration as a U.S. national standard. The Linked Systems Project completed implementation of the draft protocol it had submitted to NISO and has been operating using this protocol for a number of years, exchanging bibliographic and authority records among project participants. (Project members are upgrading their information retrieval application protocol to bring it into conformance with the version of the protocol that was ultimately standardized by NISO.) (See [8] and the references included there.)

As Committee D of NISO refined the protocol over the next few years (it was balloted in 1984 and 1987, with final adoption as a standard in 1988), it became clear that the protocol under development had far more general application than was initially conceived, and that it could be used to search a wide variety of information resources, not simply collections of bibliographic records. Furthermore, it was perceived to offer a basis for the development of information servers in a distributed environment that could be queried by clients running on workstations or mainframes, thus allowing a user to obtain access to a range of disparate information resources through a common user interface. This class of applications is now viewed as one of the most important information access problems that Z39.50 can help to solve [9, 2].

Since the Z39.50 standard was adopted in the U.S. in 1988, work on information retrieval protocols has taken place in the international standards bodies. Under ISO Technical Committee 46, Subcommittee 4, a protocol called Search and Retrieval (SR) was prepared; this is currently completing Draft International Standard (DIS) ballot as ISO 10162 (the service definition) [10] and ISO 10163 (the protocol specification) [11]. Aside from some very minor technical differences and the omission of the security and resource control functions of Z39.50 (described below), ISO 10162/10163 is essentially identical in function to NISO Z39.50. Since in the period between 1986 and 1990 the ASN.1/BER abstract and transfer syntax definitions had matured, ISO 10162/10163 replaces the ad-hoc abstract and transfer syntax definitions of Z39.50 with ASN.1/BER. It is intended that once ISO 10162/ 10163 completes DIS ballot successfully, the maintenance agency in the U.S. (the Library of Congress) will harmonize the U.S. Z39.50 standard with ISO 10162/10163, rendering Z39.50 a compatible superset of the ISO protocol.

In addition, a Z39.50 implementer's group (ZIG) was set up in the U.S. in 1989; this group meets every few months to coordinate implementation work and discuss protocol extensions, registry issues, and similar matters. About 15 major U.S. implementer or potential implementer organizations are represented in this group, which maintains a mailing list (Z3950IW@NERVM.NERDC.UFL.EDU) and an anonymous FTP server.

Two final points should be made about the development of Z39.50 as a standard. First, the requirements for an information retrieval application are enormously different than those for a distributed database application. Briefly, the most important distinction is this: A distributed database application, using a query language such as Structured Query Language (SQL), requires that clients know a great deal about what data elements are called and how they are structured into relations. For example, in a bibliographic information retrieval application, a single logical data element such as author could be fragmented into dozens of columns scattered across several interrelated tables; in order to find books by authors with a given "name" as supplied by a typical user of such a bibliographic information retrieval (IR) system, complex joins and transversals of relations would be required. It is clearly absurd, in a large-scale network environment, to expect a client to know the structure of each information resource it wishes to search at this level of detail, and to carry out this kind of processing. An IR application requires a protocol structure that can permit a client to query on logical content in the target information resource. There are many additional points on which the needs of distributed database applications and information retrieval fail to match. (For more details, see [4]. This paper also discusses differences in functional requirements between IR applications and other OSI protocols such as X.500 and FTAM, which occasionally have been suggested as rendering an IR protocol unnecessary.)

The second point concerns the debate about the OSI protocol suite versus the TCP/IP protocol suite. Although Z39.50 does make integral use of the presentation layer services and the transfer syntax of ASN.1 (in the new version), plus some use of the Applications Common Service Element (ACSE), beyond these dependencies it makes little difference whether the protocol is run over an OSI stack or a TCP/IP stack; the protocol does not rely on X.500 Directory Services, for example. As a standards activity being conducted under the auspices of national and international standards bodies (as opposed, for example, to the Internet Engineering Task Force), it made sense to cast the standard within the context of the overall OSI standards activity. The Z39.50 implementer's group has developed working agreements for running Z39.50 on top of a TCP/IP protocol stack, and most implementers currently are either working in a TCP/IP protocol environment or doing dual stack (TCP/IP and OSI) implementations. Thus, it is possible to use Z39.50 in either a TCP/IP or OSI environment.

Protocol Functions of Z39.50

Z39.50 specifies both a general framework for transmitting and managing queries and results, and a syntax for formulating queries. The framework for searching and record transfer is independent of the semantics of the information being retrieved. The query syntax is essentially parameterized by semantic data about the type of information in question. In addition, an actual IR session between a client and server assumes a base of mutual understanding about the semantics of the information being transferred. This section first reviews the "mechanical" part of Z39.50—search submission, management, and result set transfer—and then explores the more information-semantic-specific query structure and mutual understandings assumed to exist between server and client.

A server is presumed to contain a set of one or more named information resources, each of which contains a set of objects called "records." Associated with each record is a set of values for various access points. Although not all of the records within a single information resource need to contain all of the same access points and information elements, it is assumed that each information resource supports a set of access points, and that all of the records within an information resource can be placed in a common transfer format for shipment to clients. Note that I specifically avoid the term

"database" (which is used in the Z39.50 standard) to describe what I call an "information resource" on a server. One information resource might consist of several databases (such as a bibliographic database, a holdings database, and various authority files) that are interlinked in complex ways by the retrieval software into a single information resource.

When a query is processed by the server, it produces a result set that is held on the server. This result set identifies the records on the server that matched the search criteria. The Z39.50 protocol does not require that the server implement a result set by copying all matching records when creating a result set, and places no constraints requiring the server to "lock" matching records (to prevent update or deletion) until released by the client. To cope with the great variation in server capabilities in the real world, the protocol requires only that a server support a single result set, but it also supports interaction with servers that allow multiple concurrent (named) result sets or qualifying ("add-on") searches that further restrict the contents of either one or multiple result sets, or the manipulation of several result sets using the Boolean operators AND, OR, and ANDNOT. The protocol does not directly support the model implemented by some IR systems in which the server automatically creates a numbered result set for each search and retains that result, incrementing the number each time a new result set is created. To fully support the protocol as a means of access to such an IR system, it would be necessary for the Z39.50 interface software on the server to maintain a local table mapping result set names (as provided by the client) to result set numbers assigned by the IR system on the server.

A well-behaved client can be designed to keep to a minimum the number of result sets being maintained on a server at any given time, thus reducing the amount of resources that the server must commit to supporting each session. Result sets may be deleted in one of several ways: they may be overwritten by a new result when a new query from the client is processed (which always happens when the server only supports a single result set); they may be explicitly deleted by the client when no longer needed; or they may be unilaterally deleted by the server. The last case is quite important because it allows the protocol to map well onto server systems that maintain the last X result sets, or that are trying to support very large numbers of users in a public access environment with limited resources and must therefore use some type of algorithm to delete result sets when resource shortages occur. Result sets are considered to be local to a given IR session. In other words, if two sessions with the same server each create a result set called X, these result sets are distinct. Furthermore, the lifespan of a result set is considered to be no longer than the life of a single IR session. Each session starts out with no existing result sets, and any remaining result sets are deleted when a session completes. One area for future study is the possibility of adding facilities to "save" a result set into the file system on the server machine so that it persists across sessions, and to recall such saved result sets in subsequent sessions. Another area for study is the concurrent use by multiple sessions of the same result set. This is important to implementers who wish to support multithread operations (i.e., several searches outstanding) which can only be accomplished through the use of multiple logical sessions.

The design of Z39.50 is characterized by an attempt to deal pragmatically with the messy environment of information retrieval, and it takes into consideration the fact that Z39.50 server capability will be layered on top of the existing retrieval systems for many major information resources. At least in the short term, it is unlikely that organizations will completely reimplement massive production databases and retrieval systems. Z39.50 offers a great deal of flexibility so that useful features of existing information retrieval systems can be made accessible to clients through the Z39.50 protocol without requiring all servers to support such capabilities. Because of this, the protocol has a realistic flavor that I find somewhat lacking in some other OSI applications layers.

One example of this pragmatism is the recognition that unilateral deletion of result sets by a server may be necessary, as described above.

Another example of the protocol's flexibility can be found in the definition of the contents of a result set following a search. A search may produce not only a complete result set but also several different types of partial results. Thus, the server may indicate that the partial result is a proper subset of the full result set that satisfies a query, or that the partial result contains records that satisfy only some of the query criteria. Depending on the strategies for query processing employed by the server, either of these cases can occur, and some servers can distinguish between the two cases. If the server can produce one or both types of partial results, it can communicate this to the client through the protocol. The protocol places no requirement on servers to provide partial results, but if the server has partial results to offer to a client, the protocol enables the server to do so. As the various protocol functions are reviewed in detail below, other examples of this pragmatic, flexible design philosophy will become evident.

Such flexibility commands a price, however. Designing a robust client implementation that works with a wide variety of servers and effectively exploits the various capabilities that each server can offer clients will require complex programming to implement the appropriate adaptive strategy. Furthermore, the user interface issues involved in allowing the client to present the capabilities and limitations of various servers to the end user are challenging and remain as yet largely unexplained.

The basic protocol interactions between client and server are as follows. A Z39.50 session is initiated by the client through an INITIALIZE request; the server (target) answers with an INITIALIZE response. Once the session is established, the client can transmit SEARCH requests to the server. Only a single SEARCH request can be pending at any given moment.[2] When a search completes, the server notifies the client of the results of a query by sending a SEARCH response. As described above, a successful query creates a result set on the server. Transmission of records from a result set on the server back to the client is initiated by the client sending a PRESENT request; the records are returned in a PRESENT response from the server. As with a SEARCH request, only one PRESENT request can be outstanding at a time. It may require many PRESENT requests to transfer all the records of a result set back to a client. The client can dispose of result sets being held on the server during a session by sending a DELETE request; the server responds to the client with a DELETE response. A session ends when the client transmits an ACSE RELEASE request to the server (Z39.50 itself does not define a RELEASE mechanism) and receives a RELEASE response back.

In addition to these basic functions, the server may interrupt any request from the client by sending the client an ACCESS CONTROL or RESOURCE CONTROL request, suspending the currently active operation (e.g., INITIALIZE, PRESENT, or SEARCH) until the client responds back to the server with an ACCESS CONTROL response or RESOURCE CONTROL response, respectively.[3] RESOURCE CONTROL would be used, for example, by a server to notify a client that a search will take a long time or will produce a large result, and to request confirmation from

[2] An IR client may maintain several concurrent connections to the same or multiple servers, but they will not have access to each other's result sets.

[3] More precisely, transmission of a RESOURCE CONTROL or ACCESS CONTROL request by the server means that transmission of a response to the pending SEARCH, PRESENT, DELETE, or INITIALIZE request by the server is held in abeyance until the ACCESS or RESOURCE CONTROL interchange is complete. It is up to the server whether to actually suspend local processing of the SEARCH, PRESENT, etc. In the RESOURCE CONTROL function, the server can actually tell the client whether processing has been suspended at the server.

the client that the server should proceed to process the query. ACCESS CONTROL offers a general method for the server to challenge the client for a password or more elaborate authentication (e.g., based on public-key cryptographic techniques). The server can invoke such authentication based on databases being searched, specific records the client is attempting to transfer on a PRESENT, or any other criteria. Multiple RESOURCE CONTROL/RESOURCE CONTROL response and/or AUTHENTICATION/AUTHENTICATION response sequences can occur while a single SEARCH, PRESENT, or other request is in process.

A number of parameters and options available to both client and server within these protocol functions are worth discussing. This list is not exhaustive, and in particular omits coverage of some technical issues (such as the use of preferred message size and maximum record size, which allows use of a reasonable APDU size, even though a few records that might be retrieved are extremely long—longer than a normal "working" APDU size). The focus here is on capabilities and flexibility offered to server and client under the protocol.

In INITIALIZATION, the client may indicate that it wishes to use the SEARCH, PRESENT, and/or DELETE services, and that it is or is not willing to accept ACCESS CONTROL or RESOURCE CONTROL requests from the server. The server, in its response, indicates which of the services selected by the client it will support, and which of the services it will be employing that the client will accept (e.g., RESOURCE and ACCESS control).

A server that supports ACCESS CONTROL, for example, has two choices when confronted by a client that will not accept ACCESS CONTROL requests. It can refuse to enter into a session with that client, or it can fail any request from that client that provokes the server to send the client an ACCESS CONTROL challenge with an error indicator stating that a security challenge would have been issued but the client had indicated previously that it could not support such a challenge.

SEARCH requests can be submitted against one or more named databases. The client may also indicate which abstract syntax it prefers the server to use for records returned in the PRESENT response. This is specified by an abstract syntax name that is registered and is one of a set of abstract syntax names agreed upon during connection establishment (by the presentation layer protocol) for use during the session. If the server cannot supply the information in the requested abstract syntax, it will supply it in one of the other abstract syntaxes from the established set. In addition, for each database specified, the client can request that a particular set of information elements be retrieved from records matching the search criteria. Such a set of information elements is denoted by an element set name that is server-specific. This element set name allows the client to retrieve subsets of information elements (such as SHORT or FULL records, as defined by the server). Knowledge of what elements the client wants may be useful to some servers at the point when a query is being evaluated because it may change the optimal query processing strategy on the server.[4] The client may, in the SEARCH request, specify a result set name in which the result of the query is to be placed; if the result set already exists, the client can indicate that it is to be overwritten by setting an overwrite indicator. The SEARCH response sent from the server contains

[4] The element set name may also be specified on a PRESENT request, and need not match that specified on a SEARCH request. A polite client will tell the server as early as possible (e.g., in the SEARCH request) what element set it wants and only change element sets in a subsequent PRESENT request if absolutely necessary. Similarly, a well-designed server will optimize on the element set name in a SEARCH request, if present, but must be prepared to deal with a change in element set in a PRESENT request (by additional query processing, if need be).

the number of records placed in the result set and indicators giving the nature of the result set (partial or complete).

A PRESENT request essentially requests the transfer of X records starting at record Y within a specific result set. Again, the request can include an element set name for each database that might have contributed records to the named result set. A SEARCH request can include a conditional PRESENT request—for example, if the total number of records in the result set created by the SEARCH meets certain criteria (i.e., the total result is less than X records) and the query is evaluated successfully, a PRESENT request is to be automatically executed and the SEARCH response from the server will include some or all of the records matched by the query.

The conditional PRESENT request has been a controversial feature (see [3]) and criticized as complicating the protocol. The feature was developed because some of the protocol designers felt that when the protocol was used over slow networks or mapped into messaging environments, it would greatly improve service available to a client by avoiding an additional protocol exchange to obtain records. In fact, as networks have become faster, this has become less of an issue (except in message-based applications). However, the feature has proven valuable as a key element in the development of "stateless servers" by some Z39.50 implementers. These are servers that do not retain any result sets; all records matching a query are returned, along with the report of query results, in a SEARCH response that contains a "piggybacked" PRESENT.[5]

The RESOURCE CONTROL request contains indicators that the server can use to tell the client what type of partial result (if any) is available and whether the currently executing operation has been suspended pending the client's response to the RESOURCE CONTROL request.

The Z39.50 SEARCH function supports two query types. Type 0 permits a query to be passed as a text string from client to server. The protocol does not specify the contents of the string, which are established by prior agreement between client and server. Type 0 queries allow the record transfer and search management apparatus of Z39.50 to be used in conjunction with a server-specific query structure. The second type of query, type 1, is composed of a series of terms, expressed in reverse Polish (e.g., prefix) notation (RPN). The terms can be either an access point/attribute/value atom or a result set name. The operators linking terms are AND, OR, and ANDNOT. Type 1 is mandatory for conformance.

Attributes and access points are part of the context of mutual understanding that must exist between client and server for meaningful communication to occur in an IR application session. Specifically, the following items must be mutually understood between client and server:

- One or more attribute sets, which define 1) access points such as author, title, ISBN, and subject heading in a bibliographic information context, and 2) relational operators such as "access point equals," "access point contains word," and "access point contains as a last name."

- An error message set, which defines the error records that are transmitted when searches fail, and may substitute for individual records in PRESENT responses when a record cannot be sent from the server system to the client system for various reasons.

- Abstract syntaxes for records that are in the information resources on the server being searched by the client.

[5] Some other protocol extensions, currently under discussion by the implementer's group, are also needed to properly support stateless servers.

- A format for resource control records. If resource control is being used, information transferred from server to client as part of a RESOURCE CONTROL request must be defined. Some examples of resource control information are the estimated time to complete a query, the estimated number of records that will match the query, the time spent evaluating the query up to the time the RESOURCE CONTROL message was sent to the client, and the number of records matching the query up to the point that the message was sent to the client.

The first three items listed above are registered and are established as part of the context supplied by the presentation layer. Currently, ISO 10162/10163 registers an error message set and an attribute set suitable for searching bibliographic information resources; it is also planned that these will be registered nationally, via the Z39.50 maintenance agency. The implementer's group is working on other attribute sets that will be registered. The abstract syntax for records being transferred typically relies on other standards external to the protocol, although the protocol does define the abstract syntax of a base set of diagnostic records that can appear in result sets. For the transfer of bibliographic records, most implementers intend to use NISO Z39.2 as an abstract and transfer syntax; this is a general-purpose format that has been used, among other things, for MARC (machine-readable cataloging) records, which are an implementation of Z39.2 used to describe books, sound recordings, maps, and other library materials. For other types of information, where the record formats are not as well established as they are in the Z39.2/MARC standard, it is likely that ASN.1 definitions for interchange will be developed and registered.[6]

The fourth, the format for resource control messages, is not registered in ISO 10162/10163 because ISO SR omits the resource control function. The next version of Z39.50, which will reflect the more explicit approach to registry that is found in the ISO SR documents, will register at least an initial format for resource control records.

Other information must be shared between server and client; for example, for the client to send searches to the server, it needs to know the names of the information resources that are mounted on a given server. While the protocol does require that each server implement a default element set name, for effective use of a server the client should know something about the various element set names supported by the server and what they mean. Currently, there is no means within the protocol to allow the client to determine these things dynamically, although there are proposals under discussion within the Z39.50 implementers' group within the U.S. and ISO Technical Committee 46, Subcommittee 4, to supplement the protocol with an additional service called EXPLAIN. The basic idea of the EXPLAIN service is that each server would support an additional database with a fixed name (e.g., EXPLAIN or HELP) that would be searchable through a fixed, common attribute set, and would contain records both for human reading and for intermachine communication (in a fixed format) to allow a client to learn about the information resources mounted on a server, their access points, and their transfer formats. Such a facility may appear in a future version of the SR protocol, or as an addendum to the existing protocol. Clearly, however, one major goal for future protocol enhancements must be to reduce the amount of prior agreement necessary between client and server.

Authentication is another area where there must be out-of-protocol agreement between server and client. The INITIALIZE function includes a parameter for the transmission of authentication information from client to server. In addition, the ACCESS CONTROL function allows the server

[6] Even for the Z39.2/MARC format, there is some interest among implementers in developing an ASN.1 record description that can be mapped to and from the existing Z39.2/MARC format, but there are substantial questions about the best way to do this, and the entire proposal is somewhat controversial.

to challenge the client and the client to respond to the challenge. The actual details of the information that is moved between client and server in these exchanges are not specified in the protocol. Ultimately, these should be provided by some other standard with wide applicability, and are not a specific Z39.50 issue.

In addition to an EXPLAIN facility (or some other mechanism to fulfill that function), there are many other features that should eventually be made a part of the information retrieval protocol. This subject is discussed in depth in [12]. Specific areas that NISO Committee D identified as worthy of future investigation are enumerated in the preface material of the Z39.50 standard and reappear in ISO 10162/10163. These areas include the ability to save result sets across associations, the ability to sort result sets, and the ability for the client to obtain information about available access points, databases, and supported element sets (which might be handled through the proposed EXPLAIN mechanism discussed above). There is also interest in a browsing facility (which provides a list of access point values appearing in a database that are lexically close to some initial browse value supplied by the client.) The implementer's group has also identified several other areas that may require extensions, including the ability for a client to set resource control parameters at the server, a mechanism for a client to abort a search in progress, the ability to pass back, in a search response, information about stopwords, term postings and partial result sizes for terms appearing in queries, and a way for a client to determine total resource consumption to date (including charges, if applicable). Finally, although the implementer's group has come up with some initial approaches for transferring large text or image objects using Z39.50, there is clearly much more work to do in this area.

Functionally, as mentioned above, the ISO SR protocol is essentially a compatible subset of Z39.50; it omits the access control and resource control functions in its present form.

Implementation Projects—Status and Type of Applications

There are a number of Z39.50 implementation efforts now operating in prototype, although currently most of these efforts do not exactly implement Z39.50 or ISO 10162/10163. However, most of these implementations are being upgraded. Several additional projects should become operational in 1991. Most of these deal in some form or another with library automation, but a few are totally outside the field of library automation, and even within this field several projects go well beyond simply providing access to bibliographic databases.

Aside from the Linked Systems Project discussed above, other work is underway at OCLC, RLIN, the Library of Congress, the University of California Office of the President, Digital Research Associates (DRA), Pennsylvania State University, NOTIS, and the Florida State Center for Library Automation to use Z39.50 for abstracting and indexing or bibliographic databases. A number of additional institutions are also following developments carefully and planning projects, including Virginia Tech and Dartmouth. NYSERNET/PSI has worked with OCLC to develop a workstation-based client. Project Mercury, a joint effort of Carnegie Mellon University, OCLC, and Digital to develop prototype electronic libraries, is using Z39.50 to provide access not only to bibliographic databases but also to databases containing text and images. Many of the other institutions that are starting with bibliographic access applications also plan, ultimately, to provide access to textual and/or image databases. It is likely, for example, that the new electronic journal under development jointly by OCLC and the American Association for the Advancement of Science will be available through a Z39.50 interface.

DRA and the University of California are working together to use Z39.50 as a means of allowing online catalogs to extract circulation status information from a DRA system for display to the online catalog user. The Corporation for National Research Initiatives is studying Z39.50 as a method for knowbot communication within the Digital Library System architecture they are developing [13].

Dow Jones, Apple, and Thinking Machines are working together on a wide-area information server project. This project uses a version of Z39.50 that has been extended to permit the use of stateless servers and relevance feedback techniques to search Dow Jones textual databases on a Connection Machine from Macintosh clients.

The University of California at Berkeley is developing a campus-wide information server that will offer access to a wide range of nonbibliographic databases, such as course listings, job listings, university press releases, and policy and procedure manuals, based on a client-server architecture and Z39.50.

Misconceptions about Z39.50

This section comments on a few areas that seem to be sources of confusion about Z39.50.

Z39.50 and Libraries

Z39.50 is a protocol for information retrieval, not simply for bibliographic retrieval (which can be considered a specific case of information retrieval). In fact, there are other protocols that deal specifically with library applications, such as the interlibrary loan protocol (ISO 10160/10161 [14, 15]), and that are intended primarily for communication among libraries. Z39.50 and the interlibrary loan (ILL) protocol are separate and unrelated developments, although a host might well implement and use both and in fact might use Z39.50 in support of ILL.

Z39.50 is likely to be used in a wide variety of contexts, as is illustrated by the range of implementation projects already underway. It will be used by major information resource providers to move information from one system to another (as in the Linked Systems Project), and also by users who are simply searching existing information resources. The protocol design is intended to support the full range of such applications.

There is considerable confusion between Z39.50 and the MARC formats used by libraries to encode and share bibliographic descriptions of library materials. Although Z39.50 can and is used to support interchange of these records, it in no way relies on such records. These records are not intended to be directly readable by end users. Just as today's online library catalogs contain databases of records that arrived in MARC formats and provide end-user searching and display of the data, a Z39.50 client on a workstation could search such databases, transfer MARC records back to the workstation, and present them in intelligible form to an end user. Or, as has been proposed by some members of the implementer's group, a simple summary bibliographic record could easily be defined (using, say, ASN.1) for use by simpler workstation software. However, these issues are outside the scope of the Z39.50 protocol; the protocol simply provides a framework through which such applications can be implemented.

Z39.50 and OSI

Z39.50 is not based upon OSI services such as ROS (Remote Operations Services), defined in ISO 9072. There are a number of reasons for this. First of all, the designers felt that defining the protocol directly would provide more flexibility in its development over time, and that minimizing dependency on other OSI apparatus such as ROS would facilitate its rapid implementation and use in non-OSI environments (such as the Internet). In addition, there was a serious pragmatic issue: while the protocol was under development, ROS had not matured to the point where it provided a stable base for information retrieval services. The community that needed Z39.50 did not want to wait for ROS.

Most importantly for the long term, there are substantial technical problems in using ROS, at least in the context of an RDA specialization for information retrieval, which has been studied in some detail by the National Library of Canada [16]. As future versions of Z39.50 add more functions, these problems could become more serious, and it seemed better to avoid them. (This issue is discussed in more detail in [4].)

There is no requirement for the use of ROS in the development of an OSI application layer protocol; Z39.50 is a fully legitimate OSI application in its current form.

Conclusions

A protocol for information retrieval is a complex undertaking. Information retrieval systems themselves are a research area, and there is little standardization. Performing information retrieval as a network application is still relatively unexplored; there are major questions as to whether it will be possible to design a general-purpose protocol that will permit a user interface to be separated from an information retrieval server but will provide the same quality of access to the end user as a tightly integrated user interface and information retrieval system (see [12]). Z39.50 provides an important starting point for this effort, and a basis for further development. In addition, during its development the designers took into account existing large information retrieval systems, and attempted to include the features and functions necessary to accommodate these systems without major redesign as they become network servers. This is essential if the protocol is to see wide use.

We believe that Z39.50 and its future enhancements are going to be a key tool in developing the environment of networked information resources that are envisioned as a major part of the promise of the proposed National Research and Education Network.

Acknowledgements

I would like to thank Ray Denenberg and Wayne Davison for extensive discussions about the scope and contents of this paper; NISO Committee D and ISO Technical Committee 46, Subcommittee 4, Working Group 4, for discussions on the protocols discussed above; Craig Partridge for his suggestions on an earlier draft of this paper; and Mary Jean Moore for her editorial assistance.

All errors and opinions contained in this article are the author's own.

Bibliography

[1] National Information Standards Organization (NISO). *American National Standard Z39.50, Information Retrieval Service Definition and Protocol Specifications for Library Applications* (New Brunswick, NJ: Transaction Publishers, 1988).

[2] Clifford A. Lynch and Cecilia M. Preston. "Internet Access to Information Resources." In: *Annual Review of Information Science and Technology (ARIST), Volume 25* (Martha E. Williams, ed.) (Amsterdam: Elsevier Science Publishers B.V., 1990), pp. 263–312.

[3] Martin L. Schoffstall and Wengyik Yeong. "A Critique of Z39.50 Based on Implementation Experience," *Computer Communication Review* 20:2 (April 1990), pp. 22–29.

[4] Clifford A. Lynch. "Information Retrieval as a Network Application," *Library Hi-Tech* (in press).

[5] David Hartman. *Message Delivery System for the National Library and Information Service Network: General Requirements.* Prepared by the Network Technical Architecture Group. (Washington, DC: Library of Congress, Network Development Office, 1978.)

[6] Lenore S. Maruyama. *The Library of Congress Network Advisory Committee: Its First Decade.* (Washington, DC: Library of Congress, Network Development and MARC Standards Office, 1985).

[7] James S. Aagaard. *Application Level Protocol Development for Library and Information Science Applications. Volume 1: Service Definition. Volume 2: Protocol Specification.* (Washington, DC: Council on Library Resources, Inc., 1982).

[8] Judith G. Fenly and Beacher Wiggins (eds.). *The Linked Systems Project: A Networking Tool for Libraries* (Dublin, Ohio: OCLC Online Computer Library Center, Inc., 1988).

[9] Michael Buckland and Clifford A. Lynch. "National and International Implications of the Linked Systems Protocol for Online Bibliographical Systems," *Cataloguing and Classification Quarterly*, 8:3/4 (Spring 1988), pp. 15–33.

[10] International Organization for Standardization (ISO). *Search and Retrieve Service Definition.* ISO/TC46/SC4/WG4. ISO/DIS 10162.

[11] International Organization for Standardization (ISO). *Search and Retrieve Protocol Specification.* ISO/TC46/SC4/WG4. ISO/DIS 10163.

[12] Clifford A. Lynch. "The Client-Server Model in Information Retrieval." In: *Proceedings of the ASIS Mid-Year Meeting, 1989* (in press).

[13] Robert E. Kahn and Vinton G. Cerf. *An Open Architecture for a Digital Library System and a Plan for its Development. The Digital Library Project, Volume 1: The World of Knowbots.* (Washington, DC: Corporation for National Research Initiatives, 1988).

[14] International Organization for Standardization (ISO). *Interlibrary Loan Application Service Definition.* ISO/TC46/SC4/WG4. ISO/DIS 10160.

[15] International Organization for Standardization (ISO). *Interlibrary Loan Application Protocol Specification.* ISO/TC46/SC4/WG4. ISO/DIS 10161.

[16] Software Kinetics Ltd. *RDA Specialization for Bibliographic Information Retrieval* (Stittsville, Ontario: Software Kinetics Ltd., February 26, 1988).

INFORMATION RETRIEVAL AS A NETWORK APPLICATION

Clifford A. Lynch

with sidebars by **Mark Hinnebusch, Paul Evan Peters,** and **Sally McCallum**

The nature of information retrieval applications, the Z39.50 protocol, and its relationship to other OSI protocols are described. Through Z39.50 a client system views a remote server's database as an information resource, not merely a collection of data. Z39.50 allows a client to build queries in terms of logical information elements supported by the server. It also provides a framework for transmitting queries, managing results, and controlling resources. Sidebars describe the Z39.50 Implementors Group, the Z39.50 Maintenance Agency, and international standards for OSI library application protocols.

Lynch is director of the Division of Library Automation at the University of California (UC) Office of the President, where he is responsible for the UC MELVYL information system (one of the largest public access information retrieval systems in existence) as well as the computer internetwork linking the nine UC campuses. Cliff was also a key participant in the development of the Z39.50 standard.

Hinnebusch is the assistant director for Computing Services at the Florida Center for Library Automation. He is the principal designer for the FCLA Z39.50 implementation, and is the chair of the Z39.50 Implementors Group, which was formed in March 1990. *Peters* is director of the Coalition for Networked Information, and is also the chair of NISO. *McCallum* is head of the Network Development and MARC Standards Office at the Library of Congress, and is also the convenor of the working group within ISO TC46, which is responsible for OSI Application layer protocols for library applications.

Introduction and Overview

This article describes the functions of a protocol that allows the construction of "information servers"—resources attached to a computer communications network, which can be accessed by client machines to retrieve information. This concept is qualitatively different from almost all previous network applications. In this application, the communicating computers deal with information, rather than merely with data. The partner computers share an understanding of the semantics of the data that are being selected on one machine and moved to another, thus transforming that data into information.

Under the general name of Open Systems Interconnection (OSI), research and standards development over the past decades has achieved a common, consensus-based understanding among systems implementors and a growing standards and technology base that can support a wide range of distributed computing applications, including distributed databases, distributed file systems, distributed resource directories, and distributed information servers. Parallel, large-scale research and development projects such as the Internet have complemented the formal standards development process and contributed both technology and understanding of practical engineering and operational issues, further advancing the growth of distributed computing. Today, the parallel streams of applied research and development and standards are converging. In a real sense, actual adoption of all the specifics of the OSI protocol suite is not as important as the effect that its development process has had on the technology and in establishing a new paradigm for developing information

systems. Distributed computer applications will arrive in the 1990s. Only the details of implementation, and how they are to be phased into the existing, substantial computer-communications infrastructure in the United States and internationally, remain unresolved. The demand for these new applications and capabilities is urgent and widespread.

The first versions of protocols for all of the distributed computing applications listed above, and many others, are finally nearing completion. Standards work is frustratingly slow, based as it is upon consensus and performed largely by volunteer resources. And completed and adopted standards often fall considerably short of the functionality that will ultimately be needed for real-world applications, due to time and resource constraints on the standards-writing groups and, sometimes, difficulty in achieving the necessary consensus, particularly for complex functions where little prototyping has been done and the base of engineering experience is limited. Also, because of the complexity of the standards process, and the vast number of committees involved, related standards developed by different committees do not always link up as well as they need to initially. This is a particular issue in computer networking, where many interrelated standards must be used together to build functioning applications. Yet the standards are maturing to the point where they can provide an effective base for the development of distributed applications.

The OSI standards suite is an almost unprecedented effort. While most standards work traditionally considered the dominant technologies that occupy the marketplace, OSI has initiated an era of pre-emptive standards. Because of the immense cost of bringing new technologies in computing, communications, and consumer electronics to market, it is economically advantageous to establish standards first, and then implement and prove them in the marketplace; in some cases, the costs are so great that prior agreement is a virtual necessity. Vendors now often wait until the standards are at least in near-final form prior to product implementation, with only limited prototyping of early standards drafts by universities, industrial research labs, and similar R&D organizations providing validation and feedback to the standards development effort. The work on OSI application protocols is unusual for another reason. When the effort began, there was very little experience upon which to base the design of open distributed systems. The OSI work thus became a forum for discussion and agreement on models and approaches for distributing computing functions, not just an effort at codifying the specifics of inter-machine communication. This effort at understanding distributed computing, rather than just codifying it, has been a valuable result of the OSI undertaking.

The current status of the OSI application protocols is extraordinarily confusing, particularly to those who are not closely involved in the ongoing standards process. Suppose a user organization is considering how to employ an OSI application protocol for information retrieval. The protocol suite offers many options, and in coming to a decision this user organization will face the following problems:

1. In exploring the published literature and talking with people active in the OSI effort, they may be told that at least the following standards may be relevant:

   - Z39.50/ISO 10162/10163 (Search and Retrieval) for computer-to-computer information retrieval.[r1-3]
   - FTAM (File Transfer, Access, and Management) ISO 8541.[r4]
   - ISO 9594, for access to resource directories and other information resources.[r5] This is better known as X.500. As application services, these two standards are essentially the same, though they are described in different documents.
   - RDA (Remote Database Access) ISO 9579 for access to database servers.[r6-7]

   To further confuse the issue, for reasons that remain obscure to this author, the Z39.50/ISO 10162/10163 work is virtually unknown outside of the library community. It is not even mentioned in Marshall Rose's work, *The Open Book: A Practical Perspective on OSI*,[r8] which until recently was the only comprehensive tutorial work on the OSI effort. The only coverage of Z39.50 I have seen in the "mainstream" computer-communications literature is in the rather inaccurate critique by Schoffstall and Yeong,[r9] which describes Z39.50/ISO SR incorrectly as highly specialized to a narrow set of library applications, and perhaps not fully consistent with the general spirit of OSI.[n1]

2. There are basically no running production implementations of any of the listed protocols that fully match the potential of the published protocols. This situation is slowly changing with regard to available implementations. But many of these protocols have still not received serious use in the "real" applications world.

3. In many cases it is not easy to determine the published specifications for these protocols. Some protocols are still in the draft stages of the standards process. In almost all cases, the published specifications need to be supplemented with various "profiles" and "implementors' agreements," which are not obviously accessible and may still be in flux. Often, it is not easy (or inexpensive) to get even the published protocol specifications.

4. Many of the protocols involved do not incorporate all of the functionality that will ultimately be required for the intended applications. Consequently, the various protocols listed above, for example, are not yet fully differentiated, and share a particularly confusing superficial resemblance. In many cases, they appear to provide different ways for performing basically the same functions.

5. The standards documents, once obtained by the mythical evaluating organization, are often unreadable. They obscure the authors' original intentions, and lack tutorial and background material on the model underlying the protocol or on how to apply the protocol to practical problems.

This article is a modest attempt to sort out these questions for information retrieval applications. It describes particularly the functioning of the Z39.50/ISO 10162/10163 family of protocols, and considers it against other, apparently overlapping, efforts such as FTAM, X.500, and RDA. The "published" specifications, as well as areas related to implementors' agreements, will be discussed. From a personal perspective, I will also discuss the intent and conceptual models of various protocols; I have been involved with the standards committees (NISO Committee D for Z39.50 and ISO Technical Committee 46 Subcommittee 4 for Search and Retrieval) for some time. Discussions of the other protocols draw primarily on my reading of the available documents and secondary sources, and on discussions with other knowledgeable persons.

The focus of this article is primarily on how the various protocols work, what they do, and how they differ. The importance of the protocols—particularly information retrieval protocols—in various applications areas is well covered elsewhere.[10-18]

The Nature of Information Retrieval Applications

Before discussing the function of the application protocols in question, I will review what information retrieval applications actually do. A user wants information from a system based on criteria such as "articles written by John Doe," "books about the Civil War," "Bibles in Portuguese from the 17th century," or "chemicals with a boiling point above 1500 degrees." Typically, the user first wants to know how many items match the query, and it may not be entirely clear what information is contained in a given system. A query may produce no matches, or it may match hundreds of thousands of items (motivating the user to refine the query). Costs of executing a given query may be unclear; and systems may indicate that a query will take a long time (or will cost a lot of money if the system recharges for use), and then request confirmation that the query is really to be executed.

The matching process can be complicated by subtle assumptions. On some systems supporting bibliographic data, initial articles are ignored in matching titles; matching is case-insensitive and ignores certain imbedded punctuation (e.g., "data-base" matches "database"). If an author name is specified as "Tom Jones" it is assumed that "Jones" is a last name; asking for "Tom Jones" and "Jones, Tom" may, however, produce different results if the system assumes that "Tom" must be the first name in one case and is simply not a last name in the other case (e.g., the name "C. Tom Jones" might match "Tom Jones"). Some systems make allowance for variant romanization methods; others implement full authority control through an auxiliary set of information that records, for example, that Mark Twain, Samuel Clemens, and Quintius Curtis Snodgrass are all the same author. There are many different technical approaches to implementing these functions. The user is largely oblivious to the details of a specific system's technical design decisions.

The key points are that the user knows that the information in the system contains certain logical information elements; the query is expressed in terms of these elements; and the user is likely to be uncertain about what the results of the query will be.

The Common Framework

When viewed in a distributed systems context, an information retrieval application fits within what is called a client-server model. Here, the user (who might be a human being, or might be another program) interacts with software on a client machine. The software on the client machine communicates with a server, housing the information in question across the network on behalf of its end-user (human being or program). Software on the server processes the request, obtains the requisite information, and returns it to the client across the network. At no point does the actual user of the client system interact directly with the server. The application protocol defines the data that is interchanged between server and client and the rules

by which that data are exchanged. It does not address how the user interacts with the client software. Some OSI protocol specifications [including Z39.50/ISO 10162/10163] use the alternative terminology "origin" for client and "target" for server.

The OSI model also defines the environment in which all of the OSI application protocols discussed below are intended to operate. This environment has two components: the lower layers and the Application-layer service routines (the Association Control Service Element, ACSE). An Application-layer protocol can assume not only the presence of reliable two-way data transfer between communicating hosts (provided by the Transport layer), but also the data mapping services of the Presentation layer, which allow disparate machines to pass instances of datatypes such as integers from one machine to another without worrying about whether a given machine internally stores integers least- or most-significant bit first. The Presentation layer performs the necessary conversions. The Association Control Service Element establishes the necessary initial connection between Application-layer protocol implementations for a specific application.

Z39.50 and the Search and Retrieve Protocol (ISO 10162/10163)

Z39.50 is an American National Standard adopted in 1988. Work on this protocol began in the early 1980s and has its roots in even earlier protocol development in the late 1970s. It was developed by the National Information Standards Organization (NISO), the ANSI-accredited standards-making body responsible for publishing, libraries, and information services standards. ISO 10162 and 10163 (called Search and Retrieve, or SR) are the international analogs of Z39.50 and are currently in the Draft International Standards (DIS) ballot process. Although some changes will have to be made to Z39.50 once ISO 10162 and 10163 are stable (see the McCallum sidebar in this article), functionally, ISO 10162 and 10163 can be viewed as a compatible subset of Z39.50. There are two ISO standards because the ISO SR description is broken into a service specification (which defines the functions of the information retrieval protocol) and a protocol specification (which defines the contents of the actual data structures that are moved from machine to machine).

Z39.50 specifies both a general framework for transmitting and managing queries and results and a syntax for formulating queries. The search and record transfer framework is quite independent of the semantics of the information being retrieved. The query syntax is essentially parameterized by semantic data about the type of information in question. In addition, an actual IR session between a client and server assumes a base of mutual understanding about the semantics of the information being transferred. This section first reviews the "mechanical" part of Z39.50—search submission, management, and result set transfer—and then explores the more information-semantic-specific query structure and mutual understandings assumed to exist between server and client.

A server is presumed to contain a series of one or more named information resources, each of which contains a series of objects called "records," and associated with each record is a series of values for various access points. While not all of the records within a single information resource need contain all of the same access points and information elements, it is assumed that there is a set of access points supported by each information resource, and that all of the records within an information resource can be placed in a common transfer format for shipment to clients. Note that I specifically avoid the term "database" (which is used in the Z39.50 standard) to describe what I call an "information resource" on a server. One information resource might consist of a number of databases (such as a bibliographic database, a holdings database, and various authority files) that are interlinked in complex ways by the retrieval software into a single information resource.

When a query is processed by the server, it produces a result set that is held on the server. This result set identifies the records on the server that matched the search criteria. The protocol does not require that the server implement a result set by copying all matching records when creating a result set, and places no constraints requiring the server to "lock" matching records (to prevent update or deletion) until released by the client. To cope with the great variation in server capabilities in the real world, the protocol requires only that a server support a single result set, yet it supports interaction with servers that allow multiple (named) result sets or qualifying ("add-on") searches that further restrict the contents of either one or multiple result sets. The protocol does not directly support the model implemented by some IR systems where the server automatically creates a numbered result set for each search and retains that result, incrementing the number each time a new result set is created. To fully support the protocol as a means of access to such an IR system, it would be necessary for the Z39.50 interface software on the server to maintain a local table mapping result set names (as provided by the client) to result set numbers assigned by the IR system on the server. A well-behaved client can be designed to keep to a minimum the number of

SIDEBAR: HISTORY OF Z39.50 AND ISO SR

Sally H. McCallum

In 1984 a New Work Item (NWI) proposal on library application protocols was presented to ISO TC 46 (the Technical Committee responsible for library and information service standards). Prior to that time, pilot projects in computer-to-computer communication were underway in Canada, Norway, and the United States, and there was strong interest in the Netherlands.

The United States' work had focused on the Linked Systems Project (LSP), which had at that point adopted the emerging OSI model for basic communication and was experimenting with both information retrieval and record transfer applications. By 1984, the information retrieval protocol used in LSP had been sent to NISO for standardization and a NISO subcommittee was already at work on the standard that was approved in 1988—ANSI/NISO Z39.50. It should be noted that, while the LSP work formed the basis for Z39.50, the standardization process introduced many changes as it shaped the new standard for a broader constituency.

Norway had a project that transmitted information retrieval and circulation transactions. Canada had been concentrating on a protocol for interlibrary loan.

The NWI was accepted by ISO/TC 46 and a schedule for the development of needed protocols was worked out. Since the protocol work was viewed as a follow-on to data dictionary and message component standards also being developed in ISO/TC 46, coordination with the group responsible for those standards was essential. The first two protocol projects begun in TC 46 were Search and Retrieve (SR) and Interlibrary Loan (ILL).

Both of these protocols are now being balloted as Draft International Standards (DIS) in ISO: ISO 10160 and ISO 10161 are the ILL specifications; ISO 10162 and ISO 10163 are the SR documents. The DIS ballot is the final 6-month all-TC ballot that standards must pass in order to become an ISO standard.

Between 1984 and this 1990 DIS ballot, many ISO drafts were prepared, criticized, and discussed at meetings, and revised. Reaching agreement at the international level, especially when different national projects are fueling the discussion, is not an easy process. It is obviously also affected by language differences among the participants and, in this case, the complexity of these standards.

The United States' contributions and ballot responses to ISO SR were closely keyed to the specifications in Z39.50 as it took shape. The two standards, ANSI/NISO Z39.50 and ISO 10162/10163, are highly compatible where they overlap, but the NISO standard does contain more functionality. ISO SR went through two Draft Proposal (DP) ballots in ISO, the Subcommittee level ballot to which ISO standards are subjected before the DIS ballot. During the two critical DP ballots it was decided not to include access and resource control in the standard, both of which are specified in Z39.50, and record update, which the Netherlands advocated. Both of these will be investigated for future standardization either as enhancements to ISO SR or as separate standards that can be used in conjunction with SR. The absence of access and resource control is thus a major difference between the national and international standards.

The one remaining major difference between Z39.50 and ISO SR is the specification of the protocol in SR using an abstract syntax notation, ASN.1. The work on Z39.50, being slightly earlier, had developed its own abstract notation to express the "semantics" of the protocol. A transfer syntax for use in implementation was given in a Z39.50 appendix as a recommendation, but not as part of the standard. ISO SR contains an ASN.1 description of the semantics of the protocol and specifies that an implementation must support the transfer syntax defined in ISO 8825, which describes how an ASN.1 description is to be translated into a transfer syntax. A group of Z39.50 implementors has endorsed use of the ISO SR ASN.1 description and requested that the Z39.50 Maintenance Agency (see sidebar by Paul Peters) start the process to enhance the NISO standard by adding this ASN.1 description. ASN.1 would also need to be developed for the additional access and resource control functions in Z39.50.

Acceptance of the ISO SR ASN.1 description will require that the following minor differences between the two standards be reconciled:

ISO SR allows the origin to specify in a search the number of hits to be called a small set and a medium set, and then designate the record composition to be used with each. For example, fewer than 5 records retrieved might be called small and 6 to 25 records might be called medium. Full records could be requested for small retrieval sets but brief records might be requested by the origin if a medium set is found. Z39.50 does not admit differentiation of the two compositions although the medium set element does appear (but with a slightly different semantic).

ISO SR contains an additional element for the origin to designate the format for the records to be returned in response to a search. Also in a search request, ISO SR contains more flexibility for the origin to set the characteristics of the query, including use of a standardized common command language.

And, finally, the semantics and structure of the diagnostic record to be used when a search fails is fully specified in ISO SR using ASN.1. In Z39.50 the diagnostic record, with semantic and structural differences, is only a recommendation in an appendix.

result sets being maintained on a server at any given time, thus reducing the amount of resources that the server must commit to supporting each session. Result sets may be deleted in one of several ways: They may be overwritten by a new result when a new query from the client is processed (which always happens when the server only supports a single result set); they may be explicitly deleted by the client when no longer needed; or they may be unilaterally deleted by the server. The last case is quite important because it allows the protocol to "map well" onto—be effectively applied to—server systems that maintain the last X result sets, or that are trying to support very large numbers of users in a public access environment with limited resources and must therefore use some type of algorithm to delete result sets when resource shortages occur. Result sets are considered to be local to a given IR session. In other words, if each of two sessions with the same server creates a result set called X, these result sets are distinct. Furthermore, the lifespan of a result set is considered to be no longer than the life of a single IR session. Each session starts out with no existing result sets, and any remaining result sets are deleted when a session completes. One idea that has been identified as needing future study is the possibility of adding facilities to "save" a result set into the file system on the server machine so that it persists across sessions, and to recall such saved result sets in subsequent sessions.

The design of Z39.50 is characterized by an attempt to deal pragmatically with the messy environment of information retrieval and it takes into consideration the fact that Z39.50 server capability will be layered on top of the existing retrieval systems for many major information resources. At least in the short term, it is unlikely that organizations will completely re-implement massive production databases and retrieval systems. Z39.50 offers a great deal of flexibility so that useful features of existing information retrieval systems can be made accessible to clients through the Z39.50 protocol without requiring all servers to support such capabilities. Because of this, the protocol has a realistic flavor that I find somewhat lacking in some other OSI Application layers. One example of this pragmatism is the recognition that unilateral deletion of result sets by a server may be necessary, as described above. Another example of this flexibility can be found in the definition of the contents of a result set following a search. A search might not only produce a complete result set, but several different types of partial results: The server may indicate that the partial result is a proper subset of the full result set that satisfies a query, or that the partial result contains records that satisfy only some of the query criteria. Depending on the query processing strategies employed by the server, either of these cases can occur, and some servers can distinguish between the two cases. If the server can produce one or both types of partial results, it can communicate this to the client through the protocol. Yet the protocol places no requirement on servers to provide partial results. If the server has partial results to offer, the protocol enables the server to offer them to a client. As the various protocol functions are reviewed in detail below, other examples of this pragmatic, flexible design philosophy will become evident.

Such flexibility commands a price, however. Designing a robust client implementation that works with a wide variety of servers and effectively exploits the various capabilities that each server can offer clients will require a good deal of complex programming to implement the appropriate adaptive strategy. And the user interface issues involved in allowing the client to present the capabilities and limitations of various servers to the end-user are challenging and remain as yet largely unexplained.

The basic protocol interactions between client and server are as follows: A Z39.50 session is initiated by the client through an INITIALIZE request; the server (target) answers with an INITIALIZE response. Once the session is established, the client can transmit SEARCH requests to the server. Only a single SEARCH request can be pending at any given moment;[n2] when a search completes, the server notifies the client of the results of a query by sending a SEARCH response. As described above, a successful query creates a result set on the server. Transmission of records from a result set on the server back to the client is initiated by the client sending a PRESENT request; the records are returned in a PRESENT response from the server. As with SEARCH, only one PRESENT can be outstanding at a time. It may require many PRESENT requests to transfer all the records of a result set back to a client. A session is ended by transmission of a RELEASE request from client to server and receipt of a RELEASE response from the server. The client can dispose of result sets being held on the server during a session by sending a DELETE request; the server responds to the client with a DELETE response.

In addition to these basic functions, the server may interrupt any request from the client with an ACCESS CONTROL or RESOURCE CONTROL request that is sent from the server to the client, placing the currently active operation (e.g., INITIALIZE, PRESENT, or SEARCH) in suspense until the client responds back to the server with an ACCESS CONTROL response or a RESOURCE CONTROL response respectively.[n3] RESOURCE CONTROL would be used, for example, by a server to notify a client that a search

will take a long time or will produce a large result, and to request confirmation from the client that the server should proceed to process the query. ACCESS CONTROL offers a general method for the server to challenge the client for a password or more elaborate authentication (e.g., based on public key cryptographic techniques). The server can invoke such authentication based on databases being searched, specific records the client is attempting to transfer on a PRESENT, or any other criteria. Multiple RESOURCE CONTROL/RESOURCE CONTROL response and/or AUTHENTICATION/AUTHENTICATION response sequences can occur while a single SEARCH, PRESENT, or other request is in process.

A number of parameters and options available to both client and server within these protocol functions are worth discussing. This list is not exhaustive, and in particular omits coverage of some technical issues (such as the use of preferred message size and maximum record size). The focus here is really on capabilities and flexibility offered to server and client under the protocol.

In INITIALIZATION, the client may indicate that it wishes to use the SEARCH, PRESENT, and/or DELETE services, and whether or not it is willing to accept ACCESS CONTROL or RESOURCE CONTROL requests from the server. The server, in its response, indicates which of the services selected by the client it will support, and which of the services it will be employing that the client will accept (e.g., RESOURCE and ACCESS CONTROL).

A server that supports ACCESS CONTROL, for example, has a number of choices when confronted by a client that will not accept ACCESS CONTROL requests. Either it cannot enter into a session with that client, or it can fail any request from that client, which provokes the server to send the client an ACCESS CONTROL challenge with an error indicator stating that a security challenge would have been issued but the client had indicated previously that it could not support such a challenge.

SEARCH requests can be submitted against one or more named databases. In addition, for each database specified, the client can request the set of information elements to be retrieved from records matching the search criteria. Such a set of information elements is denoted by an element set name that is server-specific. This element set name allows the client to retrieve subsets of information elements (such as SHORT or FULL records, as defined by the server). Knowledge of what elements the client wants may be useful to some servers at the point when a query is being evaluated because it may change the optimal query processing strategy on the server.[84] The client may, in the SEARCH request, specify a result set name in which the result of the query is to be placed; if the result set already exists, then the client can indicate that it is to be overwritten by setting an overwrite indicator. The SEARCH response sent from the client contains the number of records placed in the result set and indicators giving the nature of the result set (partial or complete).

A PRESENT request essentially requests the transfer of X records starting at record Y within a specific result set. Again, the request can include an element set name for each database that might have contributed records to the named result set. A SEARCH request can include a conditional PRESENT request: If the total number of records in the result set created by the SEARCH meets certain criteria (i.e., total result is less than X records) and the query is evaluated successfully, then a PRESENT is to be automatically executed and the SEARCH response from the server will include some or all of the records matched by the query.

The conditional PRESENT request has been a controversial feature (see reference 9) and criticized as complicating the protocol. The feature was developed from a feeling by some of the protocol designers that, when the protocol was used over slow networks or mapped into messaging environments, it would greatly improve service available to a client by avoiding an additional protocol exchange to obtain records. In fact, as networks have become faster, this has become less of an issue (except in message-based applications); but the feature has proven very valuable as a key element in the development of "stateless servers" by some Z39.50 implementors. These are servers that do not retain any result sets; all records matching a query are returned along with the report of query results in a SEARCH response.

The RESOURCE CONTROL request contains indicators that the server can use to tell the client what type of partial result (if any) is available and whether the currently executing operation has been suspended pending the client's response to the RESOURCE CONTROL request.

The Z39.50 SEARCH function supports two query types. Type 0 permits a query to be passed as a text string from client to server. The protocol does not specify the contents of the string, which are established by prior agreement between client and server. Type 0 queries allow the record transfer and search management apparatus of Z39.50 to be used in conjunction with a server-specific query structure. The second type of query, type 1, is comprised of a series of terms, expressed in reverse polish (e.g., prefix) notation (RPN). The terms can be either an access point/attribute/value atom or a result set name. The operators that link terms are AND, OR, and AND NOT.

Attributes and access points are part of the context of mutual understanding that must exist between client and server for meaningful communication to occur in an IR application session. Specifically, the following items must be mutually understood between client and server:

- ***One or more attribute sets***, which define 1) access points such as author, title, ISBN, and subject heading in a bibliographic information context; and 2) relational operators such as access point equals, access point contains word, and access point contains a last name.

- ***An error message set***, which defines the error records that are transmitted when searches fail, and may substitute for individual records in PRESENT responses when a record cannot be sent from the server system to the client system for various reasons.

- ***Abstract syntaxes for records*** that are in the information resources on the server being searched by the client.

- ***A resource control record format***. If resource control is being used, then information transferred from server to client as part of a RESOURCE CONTROL request must be defined. Some examples of resource control information are the estimated time to complete a query; the estimated number of records that will match the query; the time spent evaluating the query up to the time the resource control message was sent to the client; and the number of records matching the query up to the point that the message was sent to the client.

The first three of these types of objects are registered and are established as part of the application context. Currently ISO 10162/10163 registers an error message set and an attribute set suitable for searching bibliographic information resources. The abstract syntax for records being transferred typically relies on other standards such as the MARC format, although there is still some disagreement within the implementor community as to how much of the MARC structure should be made visible to the Presentation layer. One option would be to have the Application layer tell the Presentation layer that it was just moving a byte string from one machine to another, with the application software on both ends knowing that the byte string was actually a MARC record encoded according to the Z39.2 standard. Another option would be to make some level of the MARC field/subfield structure visible to the Presentation layer, and to actually move the MARC data as a structured object from one machine to another through the Presentation layer. Yet another option would be to treat the MARC standard (Z39.2) as defining an actual non-ASN.1 transfer syntax. (See the Davison article in this issue concerning the Presentation layer and syntax.)

The fourth of the object types listed above, the resource control message format, is not registered in ISO 10162/10163 because ISO SR omits the RESOURCE CONTROL function. The next version of Z39.50, which will reflect the more explicit statements about registry that are found in the ISO SR documents, will register at least an initial resource control record format.

Other information must be shared between server and client; for example, for the client to send searches to the server, it needs to know the names of the information resources that are mounted on a given server. While the protocol does require that each server implement a default element set name, for effective use of a server, the client should know something about the various element set names supported by the server and what they mean. Currently, there is no means within the protocol to allow the client to determine these things dynamically, although there are proposals under discussion within the Z39.50 Implementors Group within the United States (see the Hinnebusch sidebar in this article) and ISO TC 46 SC 4 to supplement the protocol with an additional service called EXPLAIN. The basic idea of the EXPLAIN service is that each server would support an additional database with a fixed name (e.g., EXPLAIN or HELP), which would be searchable through a fixed, common attribute set, and would contain records both for human reading and for inter-machine communication (in a fixed format) to allow a client to learn about the information resources mounted on a server, their access points, and their transfer formats. Such a facility may appear in a future version of the search and retrieve protocol, or as an addendum to the existing protocol.

Authentication is another area where there must be out-of-protocol agreement between server and client. The INITIALIZE function includes a parameter for the transmission of authentication information from client to server. In addition, the ACCESS CONTROL function allows the server to challenge the client and the client to respond to the challenge. The actual details of the information that is moved between client and server in these exchanges is not specified in the protocol.

In addition to an EXPLAIN facility, there are many other features that should eventually be made a part of the information retrieval protocol. This subject is discussed in depth in my article, "The Client-Server

SIDEBAR: THE Z39.50 IMPLEMENTORS GROUP

Mark Hinnebusch

National and international standards are created in an extremely political environment. Often, interested parties already have operative mechanisms that they wish to have included in the proposed standard and often these mechanisms are somewhat incompatible. Standards, at least in the United States, do not enjoy the force of law but rather gain force solely through voluntary compliance. Given these factors, often the best a standards-making body can do is accept conflicting alternatives and promulgate them as alternative standards. A good example of this is the lowest layers of the OSI model, which include X.25 packet switching, IEEE 802.3 CSMA/CD, IEEE 802.4 Token Bus, IEEE 802.5 Token Ring, and ISDN as alternatives.

This proliferation of alternatives creates the need for "profiles." A profile is an agreement to accept only a certain set of options allowed by the relevant standards. Members of a profile group agree that when communicating with one another they will use a very specific set of parameters and will, in this way, guarantee interoperability. Of course there is no guarantee that they will be able to communicate with systems that do not conform to the profile.

The Z39.50 Implementors Group was formed in March 1990 and consists of institutions and vendors interested in developing a Z39.50 support capability. Its goal is to agree on a number of options in the underlying protocols and to resolve a number of questions that arise in the Z39.50 protocol itself. The underlying protocol options were easy to arrive at and the group quickly agreed on the appropriate set of services provided by the lower OSI layers that would be used. Most of the issues concerning the Z39.50 protocol were easily resolved. Two major issues remaining are the choice of records to return in response to a search, and modifications to the protocol to allow a server system to be "stateless" (i.e., to maintain no memory of the results of a search). All of these issues can be resolved and we expect to arrive at consensus. Interoperability testing will also be a major topic of conversation at future meetings.

Model in Information Retrieval."[r19] Specific areas that NISO Committee D identified as worthy of future investigation are enumerated in the preface material of the Z39.50 standard.

Functionally, as mentioned above, ISO SR is essentially a compatible subset of Z39.50; it omits the ACCESS CONTROL and RESOURCE CONTROL functions in its present form.

Z39.50/SR and Information Retrieval Applications

Comparing the Z39.50 protocol to the general description of information retrieval applications above illustrates a good match of functionality. Through the use of attribute sets, the Z39.50 protocol allows the client to formulate queries in terms of logical information elements in the resource supported by the server. The server, in evaluating queries, can clearly support all of the subtle functions ranging from personal name matching algorithms to authority control. The client does not need to know anything about the specifics of how the server has implemented access to an information resource. The server has functionality, in the form of RESOURCE CONTROL, which permits it to handle unreasonable searches in a rational way, by passing information back to the client and asking for further instructions. The result of evaluating a query is a count of the number of matching records; the client does not receive the result set back unless it asks for it, either as a conditional PRESENT within the SEARCH request or as an explicit PRESENT following receipt of a SEARCH result.

To be fair, the match between the protocol and the functions required in information retrieval applications is imperfect. Suppose, for example, that a search retrieves a zero result (no matches). In some current IR systems, the user interface will tell the user that the zero result occurred because a specific term in a conjunctive query did not match any records in the database (for example, because a user misspelled a search term). There is no way, within current Z39.50, to allow a server to pass this kind of information back to a client so that it can be understood.[n5] In addition, the lack of the type of EXPLAIN facility described above is a serious drawback in an environment requiring a client that can communicate with many different servers in a largely self-configuring fashion. This problem will become more severe as the number of Z39.50 client and server implementations multiply.

The match is reasonably good, however. As will be demonstrated below, Z39.50 is much better than other protocols that have been suggested as "competitors" when matched up against the real needs of information retrieval applications.

Competitive Protocols

FTAM — File Transfer, Access, and Management Protocol (ISO 8571)

FTAM is designed to read or write files or sections of files stored on servers. Files may be structured or unstructured. It does not directly support searching by combinations of values in fields within records within files (as I understand the protocol). Even if it did, it would not solve the problem of providing access to information resources unless every server structured files of a given general type of information content identically and named fields within record structures identically, which is clearly absurd. There is little to support FTAM as a reasonable competitor to Z39.50 or SR.

However, it is worth recognizing that, at least in theory, FTAM can play a valuable complementary role to Z39.50 or SR. FTAM implementations, once they reach production quality, should include facilities optimized for efficient bulk transfer of data. When very large results are obtained through SR searches, it might be more efficient to employ FTAM, rather than SR PRESENT requests, to transfer such results from server to client. In theory, the Application-layer structure of OSI should allow a client to combine multiple Application-layer services and to switch among them (see the Davison article). What remains obscure (at least to me) is exactly how the client's FTAM would identify to the server's FTAM the data to be transferred. Presumably it would be necessary to extend SR with a save result set function of some type, which would logically store the result set into the FTAM filestore on the server so that it would be available for a subsequent FTAM transfer.

It has been suggested that, while FTAM cannot replace the full SR protocol, it might well be used in conjunction with the SEARCH function of SR, allowing elimination of the PRESENT function in favor of a more general mechanism. But if we consider Z39.50 specifically (rather than SR since Z39.50 is a more fully functioned version of the IR protocol because it includes RESOURCE CONTROL and ACCESS CONTROL), there appear to be some compelling reasons for retaining the PRESENT function. It is true that FTAM can provide roughly the same level of random access to records within a result set as PRESENT can, assuming that the search result is properly structured as an FTAM file to support such access. The surrogate error record technique used by PRESENT could be used for records within the FTAM file. There are a number of major problems, however:

1. On a functional level, FTAM does not appear to include any analog of the Z39.50 RESOURCE CONTROL function. Such a function is critical if the client is to offer a high-quality interface to the end-user that can provide warning of, for example, long delays in satisfying a client request in situations where records are being presented from very slow file storage (such as tape or optical storage jukeboxes, in situations where the information server only holds on traditional disk media the indexes that point to records stored on the slower media). In the related situation where the user of the client is charged for each record sent from the server, the server may want to employ the RESOURCE CONTROL function of Z39.50 to verify a willingness to pay for transfer of a large number of records prior to performing the transfer. In cases where there is a variable royalty associated with each record, for example, or different royalty charges for use of records from different databases and the client has done a multi-database search, the client cannot tell what the charges will be simply by counting the number of records it has requested the server to present.

   It may be possible, eventually, to combine FTAM with the RESOURCE CONTROL function of Z39.50, but much more work remains to be done before this is a practical solution.

2. A second functional limitation to substituting FTAM for the Z39.50/SR PRESENT function is that FTAM does not, as I understand it, allow the analog of an element set name to transfer only selected information elements from a group of records in a result set.

3. One practical problem with substituting FTAM for SR PRESENT is more subtle. In most cases it seems likely that FTAM will be vendor-provided "utility" software coming from the operating system vendor or from a third party. In most cases, because of the intimate linkage between the SEARCH function of SR and the base IR applications code on a server, it seems likely that SR will be written as part of the applications code (though it may call upon lower-layer vendor-provided utility services). Placing an SR search result into an FTAM filestore for transfer by the client will probably mean writing a copy of the entire selected record set into the FTAM filestore on the server, which may be a very expensive process with a large result set when the client only wants to browse, for example, a few selected records, or may simply discard the result set after the user finds out how large it is. A vendor-provided

FTAM will probably not provide local system interfaces for creating files within the FTAM filestore that support "lazy" evaluation—that is, every time that FTAM needs a record from that file for transfer to a client, it would invoke some program on the server to generate the record (for example by evaluating a record pointer that is part of an SR result set). While one can imagine an FTAM implementation that permitted this procedure, it seems unlikely that such an interface will be available in the commercial FTAM implementations anytime soon. The implication is that replacing SR PRESENT with FTAM could burden IR applications with a massive performance penalty.[n6]

There are other serious performance issues in using FTAM for PRESENT. In its simplest form, an FTAM READ operation consists of four request/response sequences (resulting in a total of eight transmitted messages): 1) SELECT/OPEN request followed by SELECT/OPEN response; 2) READ request followed by BULK DATA TRANSFER; 3) TRANSFER END request followed by TRANSFER END response; and 4) CLOSE/DESELECT request followed by CLOSE/DESELECT response. The READ request and the subsequent BULK DATA TRANSFER are analogous to the PRESENT and PRESENT response. All of the other message sequences represent excess overhead. If multiple PRESENTs are desired, efficiency would obviously be improved if the READ/TRANSFER sequence could be repeated indefinitely without repeating the other sequences. However, FTAM simply will not permit this behavior if the "file transfer" class is selected—the entire set of request/response sequences must be repeated for each READ operation. If any other class is negotiated (e.g., file access), the READ/BULK DATA sequence can be repeated indefinitely without repeating each of the other sequences. Unfortunately, the other classes involve excess overhead with respect to IR. At worst, then, FTAM might require four times as many transmitted messages as the PRESENT response. The effect of this is minimized if only a single PRESENT is required per association, but becomes significant as the number of required accesses increases.[n7]

FTAM files assume a tree structure. A READ operation may specify the identifier of a node and retrieve 1) the record associated with that node, 2) all of the records contained within the subtree defined by that node, or 3) all of the records at a specified level. Without going into detail, it is obvious that this does not map well to the PRESENT function. For example, to request five records beginning at record ten, there will not, in general, be a subtree with precisely these records. There are ways to work around this problem, but none is entirely satisfactory. For example, the filename could be used to indicate the records desired. But it would then be necessary to perform the entire sequence (SELECT, OPEN, etc.) for each access. Or the record identifier could be used to encode the desired records. The problem with that approach is that it probably would not map well to vendor products.[n8]

Given these considerations, it seems most appropriate to consider FTAM as a somewhat specialized complement to Z39.50/SR for IR applications that might be used to transfer more efficiently very large result sets from server to client. FTAM, as a complete replacement for Z39.50 or SR, simply lacks the requisite functions; even as a replacement for only the PRESENT function, it falls short functionally and creates significant performance problems. (See also the sidebar on FTAM in Wayne Davison's article in this issue.)

Directory Services (ISO 9594 or X.500)

OSI directory services are in use (on a prototype basis) to support applications such as "white pages" telephone directories. These applications share with SR a focus on information rather than data. It is possible to extend the directory with user-defined object types and attributes. (See the article in this issue by Daniela Planka for more information.)

The obvious major problems with using OSI directory services for information retrieval are the lack of any type of resource control function (which is essential for IR applications, as discussed previously), and the fact that a search returns all matching objects, rather than simply a count of the objects matching the query and giving the client the option to then request transfer of some or all of these matching objects. There also does not seem to be any way to retain a result set for use in a subsequent qualifying search.

Perhaps the best way to sum up the potential of OSI directory services for IR applications is that currently some of the functions needed for IR can be fit within the directory services model, but some that seem critical to me cannot. It seems likely that, as additional functions are added to the IR protocol, the divergence with directory services will grow larger, and that the objectives of the two protocols are really different.

Remote Database Access (ISO 9597)

To consider RDA as a competitor to Z39.50 requires clarification of the definition(s) of "databases." As used by the database community, a database is a collection of relations, in the relational model; as used

by the information retrieval community, a database is an information resource. They are different concepts, approached at different levels of abstraction. In an information resource, the access interface includes substantial semantics: One asks for things such as authors and titles. In a database management system one asks for fields (e.g., columns) meeting certain criteria; all semantic meaning is supplied by applications software that exists above the level of the database management system (DBMS) interface.

Further confusing matters, the remote database access service is defined in two parts. Part one, the generic remote database access service, provides a framework for the development of specializations that constitute actual RDA protocol specifications. At present there is one specialization defined, for SQL (structured query language); this is defined in Part 2 of ISO 9597. The National Library of Canada[22-23] has conducted studies on the feasibility of developing a specialization for IR use that could carry, for example, Z39.50 type 1 queries. Consequently, two separate questions need to be addressed: 1) What is the feasibility of using the existing SQL specialization of RDA for IR applications? 2) What is the feasibility of using a potential IR specialization of RDA for IR applications? The second question is more speculative since all the details of such a specialization have not really been worked out.

There has been considerable controversy about how well SQL-based relational database management systems (RDBMS) work as a platform for building various types of IR systems (see reference 20 and the additional citations it contains). There is considerable evidence that such relational systems will need substantial extension and refinement if they are to be effective in this role. However, this argument should not be confused with the question of whether the SQL specialization of RDA is in fact a suitable IR protocol. It is not, for the simple reason that it is the wrong level of interface—it deals with data, not information. All of the various deficiencies of SQL as a language for formulating IR-type queries are secondary to this point.

Consider the following scenario: One wishes to retrieve information from a bibliographic database based on the criterion that the title contains a certain word. Using the bibliographic attribute set of Z39.50/SR, this is exactly how the query is formulated. To formulate the query in SQL, even before addressing the fact that SQL does not include a "contains-keyword" operator, one must begin by establishing the name of the relation in the database on the server that contains a column for titles, and in that relation what the name of the relevant column is. One server might call the relation MONOGRAPHS, and the title column TI; another might call the relation BOOKS and the title column TITLE. Unless each client knows every detail of how the semantics of information are mapped into relations on every server with which that client might ever want to communicate, there is no hope of ever establishing useful IR application sessions. This is the critical flaw in the use of SQL as the basis for an IR protocol. SQL serves a different purpose.

An IR specialization of RDA is a much more complex matter to evaluate. Clearly, by replacing SQL with something like the Z39.50/SR type 1 query, one can achieve the appropriate level of abstraction in specifying information to be retrieved from the information resource on the server. One can define some additional commands and a result set model on the server that basically mirror those already found in Z39.50 and SR. The major source of problems arises from those functions within Z39.50 that are initiated by the server rather than the client (e.g., RESOURCE CONTROL and ACCESS CONTROL). The model for generic RDA, as I understand it, calls for all operations to be initiated by the client and responded to by the server.

The National Library of Canada's study circumvents this problem by proposing the establishment of subconnections for reverse direction communication. The overhead and complexity involved in this solution is unattractive. Further, because the client-server interaction sequences may become more complex as the IR service evolves, the restriction imposed by the RDA generic model may continue to be a problem.

To be fair, RDA offers a considerable amount of extra function that is not part of Z39.50/SR, such as concurrency control, crash recovery, rollback, and related services that are needed in a distributed, transaction-oriented environment (such as distributed SQL databases). It is not clear how important these functions are in supporting IR services, other than in the specialized contexts where IR is used in conjunction with some type of updating or transactional service. In these types of applications, use of the SQL specialization of RDA or some new IR specialization of RDA may be appropriate. In distributed transactional environments, it may be more reasonable to expect the structure of various databases to be coordinated in detail, and for clients to have detailed knowledge of the database structure of remote servers, not just of the logical content of information stored on these servers.

From a practical point of view, it should also be noted that Z39.50 has been a United States national standard for two years and that ISO 10162/10163 is currently in Draft International Standards (DIS) Ballot. The work on generic RDA is still in the Draft Proposal stage (voting closed 15 July 1990 for the most recent draft), and the current schedule (which may be optimis-

SIDEBAR: Z39.50 MAINTENANCE AGENCY: THE LIBRARY OF CONGRESS

Paul Evan Peters

Standards development organizations establish maintenance agencies to organize experiences with and recommendations concerning the implementation and evolution of particularly dynamic standards. Maintenance agencies track the identities, approaches, and accomplishments of implementors of such standards and support the efforts of those implementors in a variety of ways during the period in which implementation activities are at a critical peak. These agencies are especially important to standards development organizations, like the National Information Standards Organization, which promulgate standards by forming task committees to achieve a single result in a specific area rather than by forming standing committees to achieve a variety of results in a general area.

The National Information Standards Organization has designated the Library of Congress as the Maintenance Agency for Z39.50. The overall role that the Library of Congress will play vis-a-vis the implementation and evolution of Z39.50 is typical of the responsibilities of maintenance agencies in general:

- compiling a registry of implementors of Z39.50 and, when appropriate, making information derived from that registry available to interested parties;
- coordinating modifications of and enhancements to Z39.50 to address, among other things, defects, new functions and features, related specifications, and architectural reference models;
- coordinating preparation and dissemination of technical reports addressing such topics as Z39.50 implementation guidelines, configuration parameters, and functional profiles; and,
- coordinating development and representation of contributions by and positions of the United States on related international standards.

The current plan of action of the Z39.50 Maintenance Agency at the Library of Congress translates these general responsibilities into a program of work that is based on both an assessment of contemporary circumstances and a projection of emerging requirements. This plan provides for the:

- refinement and modification of Z39.50 to achieve compatibility with the "search and retrieval" protocol designated as ISO DIS 10163;
- re-specification of Z39.50 in accordance with the formalities of "Abstract Syntax Notation #1" (ASN.1), a language for the comprehensive and unambiguous specification of open systems interconnection protocols;
- development of a "Protocol Implementation Conformance Statement" (PICS) proforma for Z39.50 to provide the mechanism for implementors to record the details of their individual realizations of the standard so that customers compare those details to their function and feature requirements; and,
- formulation of design conceptualizations of possible new Z39.50 services such as "explain" to list databases, access points, etc. and "browse," "sort," and "save" operations on retrieval sets.

It is important to note that the efforts of maintenance agencies in general and those of the Z39.50 Maintenance Agency at the Library of Congress in particular are intended to organize and accelerate rather than to replace the process by which a specific standard is revised and approved as an American National Standard. This is to say that the due process and voting requirements enforced by all committees and organizations operating under the procedures of the American National Standards Institute, the National Information Standards Organization in particular, apply to work originating in maintenance agencies to the same degree as they apply to work originating in task and standing committees. Consequently, supplements to and extensions of Z39.50 will be approved by the Voting Members of the National Information Standards Organization in the same manner that Z39.50 and all other Z39 standards were originally approved.

Further information about the Z39.50 Maintenance Agency at the Library of Congress can be obtained from Ray Denenberg of the MARC Standards and Network Development Office at the Library of Congress or from Patricia Harris of the National Information Standards Organization.

tic) calls for progression to DIS by the summer of 1991. Distributed IR systems need to be implemented, and a standard is needed to facilitate the process. In the absence of a compelling case for using a specialization of RDA for IR, which could easily delay availability of the IR specialization in a stable form until 1992 or later, it is sensible to work from the existing, stable Z39.50/SR standard.

CONCLUSIONS

The Z39.50 and SR protocols offer a basic set of essential functions to support IR applications. Z39.50 includes one key function—resource control—which is not yet part of SR and which is essential for a large class of distributed IR applications. A great deal of additional work is needed to enrich the functions available in these protocols, both to permit new IR operations to be carried out through the protocols and to permit practical scale-up in the number of communicating clients and servers through the exchange of "meta-data" (e.g., data about the contents of information resources); but the basic protocols are sound and extensible. In addition, the Z39.50 and SR protocols are applicable, through the definition of new registered objects such as abstract record syntaxes and attribute sets, to support information resources containing a wide variety of types of information, not just bibliographic records.

This paper has also examined a number of OSI Application-layer protocols with apparently related functions, specifically, FTAM, X.500, and RDA. FTAM clearly complements Z39.50 and SR, but it cannot supplant them. And FTAM and Z39.50/SR cannot yet coordinate effectively in situations where both are needed. X.500, while superficially similar to Z39.50 and SR, lacks certain features and includes a lot of irrelevant complexity, such as support for update and for fully distributed directories. RDA can be related to SR and to Z39.50 in two ways: by considering the existing SQL specialization of RDA, and by considering a possible IR specialization of RDA. SQL, which deals with data rather than information retrieval, is clearly not the level of interface for network IR applications. An IR specialization of RDA seems to have relatively little to recommend it over the current Z39.50/SR functions, and presents problems in critical areas such as support of the resource control function. It does seem likely that the SQL specialization of RDA will have important roles to play in the construction of various types of distributed database systems, including those that support various types of library automation functions. It is even possible that an information server might be constructed that would run distributed across a number of machines, and which would be accessed externally by Z39.50 or SR but internally would use the SQL specialization of RDA to communicate from one machine to another within the "system."

As the sidebars to this article and other articles in this issue indicate, Z39.50/SR is quickly moving from theory to practice. It is clear that the protocol will evolve and be refined based on actual experience and in response to the need of additional functions. But, at this point, it seems clear that Z39.50/SR is the basis upon which protocols for information retrieval applications, and for the development of information servers, will proceed.

ACKNOWLEDGMENTS

I would like to thank Nancy Gusack for her editorial assistance, Cecilia M. Preston for her comments on an earlier draft, and Ray Denenberg at the Library of Congress for his thorough and enthusiastic review of earlier drafts, and for numerous suggestions that improved this paper. I would also like to thank my colleagues in NISO Committee D and ISO TC46 SC4 and the Z39.50 Implementors Group for discussions over the past few years that have played an essential role in developing the perspective presented here.

NOTES[n]

n1. A rebuttal article will appear in *Computer Communications Review* shortly. The interested reader should also see the letters to the editor in *Computer Communications Review* 20:3.

n2. An IR client may maintain several concurrent connections to the same or multiple servers, but they will not have access to each other's result sets.

n3. More precisely, transmission of a RESOURCE CONTROL or ACCESS CONTROL request by the server means that transmission of a response to the pending SEARCH, PRESENT, DELETE, or INITIALIZE request by the server is held in abeyance until the ACCESS or RESOURCE CONTROL interchange is complete. It is up to the server whether to actually suspend local processing of the SEARCH, PRESENT, and so on. In the RESOURCE CONTROL function, the server can actually tell the client whether processing has been suspended at the server.

n4. The element set name may also be specified on a PRESENT request, and need not match that specified on a SEARCH request. A polite client will tell the server as early as possible (e.g., in SEARCH) what element set it wants and only change element sets on a subsequent PRESENT if absolutely necessary. Similarly, a well-designed server will optimize on the element set name in a SEARCH, if included, but must be prepared to deal with a change in element set on a PRESENT (by additional query processing, if need be).

n5. The Z39.50 Implementors Group is currently discussing protocol extension to address this shortcoming.

n6. While it is difficult to quantify the size of this performance penalty, some reasoning by analogy can be made with relational database systems, which normally insist on materializing all query results even when the user just wants a count of the number of tuples in the result prior to doing anything else. Such an approach can increase I/O processing in IR applications by a factor of about 10.[r20-21]

n7. Ray Denenberg, personal communication.

n8. Denenberg, personal communication.

REFERENCES[r]

r1. National Information Standards Organization (NISO). *American National Standard Z39.50, Information Retrieval Service Definition and Protocol Specifications for Library Applications*. New Brunswick, NJ: Transaction Publishers, 1988. [Available from: NISO Editor, Transaction Publishers, New Brunswick, NJ 08903.]

r2. International Organization for Standardization (ISO). *Documentation—Search and Retrieve Service Definition. ISO/TC45/SC4/WG4. ISO/DIS 10162.* Vienna, VA: Omnicom, 1990. [Available from: Omnicom Information Service, 1115 Park St., SE, Vienna, VA 22180.]

r3. International Organization for Standardization (ISO). *Documentation—Search and Retrieve Protocol Specification. ISO/TC45/SC4/WG4. ISO/DIS 10163.* Vienna, VA: Omnicom, 1990. [Available from: Omnicom Information Service, 1115 Park St., SE, Vienna, VA 22180.]

r4. International Organization for Standardization (ISO). *Information Processing Systems—File Transfer, Access, and Management. International Organization for Standardization and International Electrotechnical Committee, April, 1988. Final Text of Draft International Standard 8571.*

r5. International Organization for Standardization (ISO). *Information Processing Systems—Open Systems Interconnection—The Directory—Overview of Concepts, Models, and Service. International Organization for Standardization and International Electrotechnical Committee, December 1988. International Standard 9594-1.*

r6. International Organization for Standardization (ISO). *Information Technology—Database Languages—Remote Database Access—Part 1: Generic Model, Service and Protocol. ISO/IECDP 9579-1.* Geneva: Switzerland: International Organization for Standardization, 29 March 1990.

r7. International Organization for Standardization (ISO). *Information Technology—Database Languages—Remote Database Access—Part 2: SQL Specialization. ISO/IEC 1st DP 9579-2.* Geneva, Switzerland: International Organization for Standardization, no date.

r8. Rose, Marshall. *The Open Book: A Practical Perspective on OSI*. Englewood Cliffs, NJ: Prentice Hall, Inc., 1990.

r9. Schoffstall, Martin L., and Wengyik Yeong. "A Critique of Z39.50 Based on Implementation Experience." *Computer Communication Review* 20:2 (April 1990): 22-29.

r10. Fenly, Judith G., and Beacher Wiggins, eds. *The Linked Systems Project: A Networking Tool for Libraries*. Dublin, OH: OCLC Online Computer Library Center, Inc., 1988.

r11. Lynch, Clifford A. "The Linked Systems Project: A Networking Tool for Libraries." (book review) *JASIS* 41:4 (June 1990): 305-306.

r12. Lynch, Clifford A., and Cecilia M. Preston. "Internet Access to Information Resources." *Annual Review of Information Science and Technology (ARIST)* 25 (1990): 263-312.

r13. Lynch, Clifford A. "Library Automation and the National Research Network." *EDUCOM Review* 24:3 (Fall 1989): 21-26.

r14. Davison, Wayne E. "The WLN/RLG/LC Linked Systems Project." *Information Technology and Libraries* 2:1 (March 1983): 34-46.

r15. Denenberg, Ray. "Linked Systems Project, Part 2: Standard Network Connection." *Library Hi Tech* 3:1, consecutive issue 10 (1985): 71-79.

r16. McCallum, Sally H. "Linked Systems Project, Part 1: Authorities Implementation." *Library Hi Tech* 3:1, consecutive issue 10 (1985): 61-68.

r17. McCallum, Sally H. "Linked Systems Project in the United States." *IFLA Journal* 11:4 (November 1985): 313-324.

r18. Lynch, Clifford A. "Access Technology for Network Information Resources." *CAUSE/EFFECT* 13:2 (Summer 1990): 15-20.

r19. Lynch, Clifford A. "The Client-Server Model in Information Retrieval." Presented at the Proceedings of the 1989 ASIS Mid-Year Meeting, May 21-24, 1989, San Diego, CA. To appear in: *Interfaces for Information Retrieval*. Westport, CT: Greenwood Press.

r20. Lynch, Clifford A. "Extending Relational Database Management Systems for Information Retrieval Applications." Ph.D. diss. Berkeley, CA: University of California, Department of Electrical Engineering and Computer Sciences, 1987.

r21. Lynch, Clifford A. "Nonmaterialized Relations and Support of Information Retrieval Applications by Relational Database Systems." Accepted for publication by *JASIS*.

r22. Software Kinetics Ltd. *Comparative Study of the ISO Remote Database Access and NISO Information Retrieval Protocols (Final Draft)*. Stittsville, Ontario: Software Kinetics, Ltd., 2 February 1988.

r23. Software Kinetics Ltd. *RDA Specialization for Bibliographic Information Retrieval*. Stittsville, Ontario: Software Kinetics, Ltd., 26 February 1988.

(Editorial, continued from page 5)

I want to express my special thanks to 1) Ray Denenberg, special issue editor, for his commitment, effort, and outstanding accomplishment in bringing this "textbook" to reality; 2) the Library of Congress for its national and international leadership in promoting OSI standards for the library community, and specifically for allowing Ray Denenberg the opportunity to prepare this issue; 3) the National Library of Canada, for its national and international leadership efforts to develop and promote OSI standards and test facilities, and for its numerous contributions to this special issue; and 4) to the many individual authors who contributed important articles and sidebars to this publication. Without the foresight, commitment, capabilities, and selflessness of institutions and persons such as these, we would not be the contemporary beneficiaries of OSI, nor would the library community have this definitive text on OSI.

On behalf of the entire library community, I thank you—the contributors to this special issue—for your past and present efforts to develop OSI standards, applications, and test facilities; and extend encouragement and best wishes as you continue to make it possible for libraries to embrace an exciting future—a future of open systems interconnection.

C. Edward Wall

Session III: Wide Area Information Server Development at UNC-Chapel Hill

J. Fulton

WAIS Development at UNC-Chapel Hill

Jim Fullton, UNC-Chapel Hill

I. History of WAIS at UNC

A. Our original needs

1. Grants Information
   - we needed a better mechanism than VTX to distribute grant information
   - client/server model based on Z39.50 seemed ideal.
2. Campus Information System
   - Wanted something more flexible than VTX, that worked in users' preferred graphical environment.
   - Needed to use our current VTX information and work in parallel with VTX.

B. Our first solutions

1. Developed simple X-based, DOS client and free-text server to manage documents.
2. Advantage: we did it.
3. Disadvantage: we did it.

C. The real solution: WAIS

1. after doing all of this work, we discovered WAIS
2. Advantages:
   - same functionality, different mechanism.
   - already in use by others
   - developed by a vendor
   - source-code form
3. Disadvantage: only worked on Mac and Unix.
4. We dumped our work and hopped on the WAIS bandwagon.

II. Software Development

A. Our goals:

1. to make initial ports to as many systems as possible so that the development community can produce clean implementations.

B. What we have done:

1. VMS server/client port for Process Software TCP/IP
2. MS-DOS clients for Waterloo TCP, Novell LAN Workplace
3. Preliminary port to MS Windows
4. SunView client
5. Preliminary port to VM

C. Standard servers

1. Newsgroup server on lambada.oit.unc.edu
2. User Services server on next1.oit.unc.edu

D. **Special servers**
   1. Astronomical Image server.
      - uses FITS data type to perform search and retrieval operations on FITS (Flexible Image Transport System) format images.
      - links images and documents together through free-text and relational databases, with WAIS access.
   2. "Movie" server
      - provides access to NeXT format "movies" through searches of descriptive documents. The "movie" is returned as the result.

E. **Special Uses**
   1. User support system
      - developed by Roger Akers at UNC as a mechanism to ease the workload on user service personnel.

III. Future Development and Design Issues

A. How will WAIS interact with other systems such as Gopher, Archie, Prospero, WWW, etc?

B. How will the WAIS protocol be standardized?

C. How will vendors and the "public" work with each other to ensure the standardization and continued evolution of WAIS?

IV. General discussion.

Session IV: The Kudzu Project in North Carolina

G. Brett

The NC Kudzu Project

George Brett, MCNC & UNC-General Administration

"a synergistic initiative that is a cooperative and collaborative paradigm shift towards an integrated coalition for open interoperable infrastructure"

I. The WAIS Support Center - the Initiative

- **provide central resource for WAIS service and support**
- **promote research and development of WAIS**
- **coordinate emerging standards for WAIS**
  - improve conformance to Z39.50
  - file descriptors
  - improve indexes
- **promote transfer of technology from research environment to real world**

II. WAIS & related activities in North Carolina

- **Center for Communications - the WAIS Center**
  - CNI Z39.50 interoperability testbed project
  - KUDZU-L
    - listserv for WAIS with regional focus
- **East Carolina University**
  - Investigating WAIS in campus wide information delivery
  - Investigating WAIS for multi-media delivery
- **North Carolina State University**
  - Project Eos
    - Help Desk
    - Manuals
  - Hooks into VTX videotext databases at 6 UNC campuses
- **North Carolina Supercomputing Center -- AVS Group**
  - provide custom interface to locate AVS program modules
    - WAIS will be internal to AVS
      - controlled access to specific resource
      - transparent to the user
    - WAIS will be also an external object/module
  - deliver all forms of AVS multimedia - modules, images, sounds, etc.
  - archive of AVS documents -- help documents, change info, tech manuals, etc. etc.

- **UNC-Chapel Hill -- Kudzu city**
  - 30+ databases mounted and available
  - servers for UNIX based workstations and DEC VMS
  - clients: MSDOS command line and Windows 3.0 (later)
  - Some examples of databases
    - Frequently Asked Questions for various environments (Mac, DOS, Unix)
    - Job listings
    - Campus Directory
- **UNC-Charlotte -- WAIS & VTLS**
  - co-development with VTLS to make their library automation software WAIS compliant

CENTER FOR COMMUNICATIONS

PO Box 12889 3021 Cornwallis Road

Research Triangle Park, NC 27709-2889

919-248-1800